Cryptography

CISC 3325 — CUNY Brooklyn College
Lecture Notes

Cryptography

In this chapter, we will:

1. Establish definitions of concepts e.g., Cryptography, Encryption, Decryption, Symmetric Key, Asymmetric Key, Hash Functions, etc.
2. Learn the history of cryptography.
3. Analyze Symmetric Key and Asymmetric Key cryptographies and explain the difference between the two.
4. Find out where hash functions are useful in the world of cryptography.

Cryptography

In the previous chapters, we learned that one way to prevent an attacker from accessing your confidential information is to store this information encrypted. But what is encryption and cryptography, and how can we (or businesses whose services we use) achieve this?

Encryption (also: **enciphering**) is the action of hiding your data by replacing it with code called **cypher text** or **cipher text** (consisting of some randomly scrambled letters.)

Cryptography is a compound of two Greek words: *crypto* means 'hidden' or 'secret', and *grapho* means 'write'. That is, cryptography is about creating code that can't be easily understood.

Decryption (also: **deciphering**) is the opposite of encryption: it is the action of converting the cipher text back into **plain text** (your original data/information.)

Cryptography

The encryption of data is done with a **key**, which is a description of how data is converted from plain text form into a cipher text form. Encryption algorithms today rely on math properties, so keys in such cases are numbers (integers). Similarly, decryption is also done with a key. Ideally, one can convert a cipher text back to its plain text form only if he or she has the decryption key.

Why do we say 'ideally'? Because data that was encrypted using a weak encryption algorithm might be attacked and revealed by a hacker, even when the hacker doesn't have the key. This is why encrypting your data with a strong encryption algorithm is critical!

A **cryptosystem** is a collection of cryptographic algorithms that implement a particular security service (a system for encryption and decryption). Typically, it consists of three algorithms: key generation, encryption, and decryption.

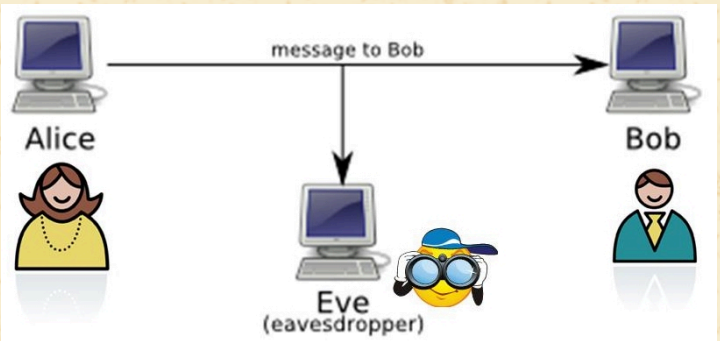
A **cipher** (also: **cypher**) is part of a cryptosystem. It consists of a pair of algorithms: one for encryption and one for decryption.

Cryptography

The main **goals** of cryptography are to:

- Hide data stored in a system/database from thieves or hackers of this database.
- Establish confidential, untampered, and unforged communication between two people or programs.

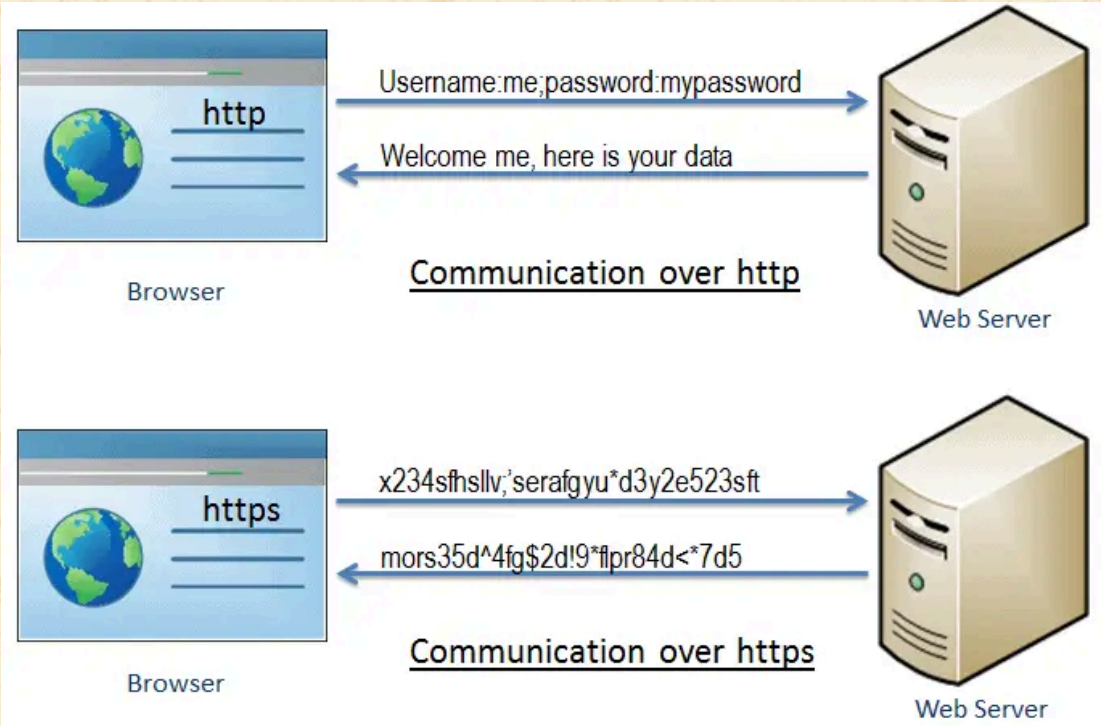
Eavesdropping is the action of tapping on an ongoing communication between two parties. The goal of an eavesdropper is to illegally listen to and obtain data that is supposed to be communicated confidentially. However, when data is encrypted during transfer, an eavesdropper will have no way to understand what those data mean.



Eve eavesdrops on Alice and Bob. Taken from these lecture notes on Encryption and Security.

Cryptography

As we said, communication between not only two individuals but also two web applications must be kept confidential:



Unencrypted (top) vs. encrypted (bottom) web communication. Taken from this HTTP tutorial.

History of Cryptography

The idea of using cryptography has existed for hundreds of years!

- One of the first pieces of evidence of using cryptography was found on the tomb of an Ancient Egyptian nobleman back in 1900 BC, not to hide information but to beatify and dignify text.
- Around 100 BC, Julius Caesar used encryption to ensure that his soldiers were the only ones who could understand messages to his army generals. The encryption system he used is called a **substitution cipher** (also: **Caesar cipher**); we'll explain more in the next slides.
- In the 16th century, the French cryptologist Blaise de Vigenère described an encryption algorithm called the **Vigenere Cipher** that uses a key. More on this in the upcoming slides.
- In the 1920s - 1970s, Enigma machines created complex letter substitutions that change with every key press of the machine.
- The 20th century saw the creation of the advanced ciphering standard DES and, in 1997, a far more secure AES standard, which is used nowadays.

History of Cryptography

Video on using cryptography in ancient times:

https://www.youtube.com/watch?v=9pp9YpginNg

Caesar Cipher

Encryption here goes as follows: take the plain text, and shift every letter in it by a constant number of letters. For example, if a shift size is 3, then every instance of the letter A will be replaced D, the letter B will be replaced by E, etc., in a circular manner (Z will be replaced by C.)

To decrypt such text, one needs to know that the shift size is 3, and simply perform the opposite operation: 'subtract' 3 from every letter (A will be replaced by X, B will be replaced by Y, etc.,) which will return the plain text.

For example, given the cipher text: `uryyb jbeyq`, if you know that the original shift size was 13, you can decrypt this text back to plain text by subtracting 13 from each letter, thereby getting: `hello world`.

Because there are only 26 letters in the English alphabet, there are only 26 possible shift sizes. This means anyone is able to manually check against each one of the shifts within a few minutes to 'crack' the plain text even without being told what the shift size is. As you might guess, it means that the Caesar Cipher is weak.

Caesar Cipher

Taken from this CodePen snippet.

Substitution Cipher

A more sophisticated cipher is one that replaces the letters of a plain text by any permutation of the English alphabet.

- First, mix the English alphabet letters in any order, and
- Second, replace A by the 1st letter in that order, B by the 2nd letter, etc.

Example: a random permutation of the alphabet is: `IQCYTWDVJLGEZHAMNKUPRSFBOX`, so every instance of A will be replaced by I, B will be replaced by Q, C will be replaced by C, etc.

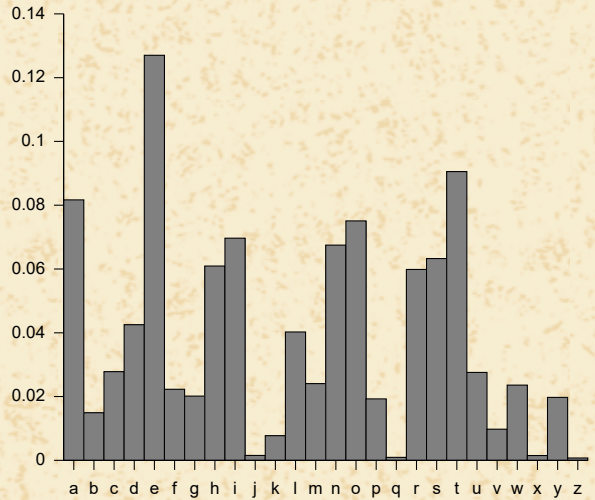
Fun Class Activity: How many permutations exist? Hint: choose one of 26 letters for the 1st place, one of the remaining 25 letters for the 2nd place, one of the remaining 24 letters for the 3rd place, etc.

The number of permutations, as the above exercise shows, is insanely large. However, a substitution cipher has a significant weakness: some English letters are more frequent than others, so one can perform a **frequency analysis** to crack the encryption.

Substitution Cipher: Weakness

Because some English letters are more frequent/common in written text than others, text that was encrypted using a substitution cipher can be converted back to plain text with relative ease, without knowing the actual permutation.

How? Scientists have computed how frequent each letter is, and came up with the following data:



English letter frequencies, sorted alphabetically. Taken from the Wikimedia Commons.

Substitution Cipher: Weakness

To attack a substitution cipher, you would note that the letter E is the most common written English letter. You will look into the cipher text for the **most common** letter, which you will undoubtedly know should be replaced by E.

Next, you'll notice that T is the next most common written English letter, so you'll look into the 2nd most common letter in the cipher text, and replace it with T.

This procedure of gradually replacing cipher text with plain text letters is called **frequency analysis**, which renders the substitution cipher weak and, therefore, insecure.

Continuing this way, you will replace many cipher text letters with plain text and will start revealing the plain text!

To make this word even easier, you may also want to keep in mind the frequency of English words, such as "I", "the", "a", "an", etc., common letter pairs, such as "th", "an", "in", "qu", and common patterns like "tion" (as in "nation") and "ing" (as in "doing").

Substitution Cipher: Weakness

Fun Class Activity: Decrypt the following:

```
Jvl mlwclk yr jvl owmwez twp yusl w zyduo
pjdcluj mqil zydkplmr. Hdj jvlz tykilc vwkc jy
mlwku jvl wkj yr vwsiquo, tvqsv vlmflc mlwc
jvlg jy oklwjulpp. Zyd vwnl jvl fyjlujqwm jy cy
jvl pwgl. Zydk plsklj fwpptykc qp: JYWPJ
```

Interesting observations: what frequent English word could `jvl` be hiding? (Note how frequent it is!)
Same about `yr` and `jy`.

You can use the following website to count the letter frequency in the above text to assist you with the frequency analysis: https://www.dcode.fr/frequency-analysis.

Substitution Cipher: Weakness

Any questions?

History of Cryptography

A stronger alternative to Caesar Cipher: Vigenère Cipher.

https://www.youtube.com/watch?v=aHSt_IzRIxw

Vigenère Cipher

Blaise de Vigenère, a 16th-century French diplomat and cryptographer, described a cipher that uses a combination of several Caesar ciphers.

- Instead of shifting each letter of your data by a fixed number, each letter will be shifted by a different number, using a fixed keyword.
- For example, if your data is `hello world`, and the keyword is `cisc`, you'll add `c = 2` to `h` to get `j`, add `i = 8` to `e` to get `m`, add `s = 18` to `l` to get `d`, add `c = 2` to `l` to get `n`, and repeat this for as many copies of the `cisc` keyword as needed. The end result is the cipher text: `jmdnq egtnl`.

Data	h	e	l	l	o		w	o	r	l	d
Keyword	c	i	s	c	c		i	s	c	c	i
Result	j	m	d	n	q		e	g	t	n	l

Vigenère Cipher: Vigenère Square (= Tabula Recta)

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
b	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a
c	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b
d	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
e	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
f	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
g	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f
h	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g
i	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h
j	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i
k	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j
l	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k
m	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
n	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m
o	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n
p	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
q	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
r	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
s	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
t	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
u	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
v	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
w	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
x	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
y	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
z	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Vigenère Cipher

As the video we watched several slides earlier informed us, the Vigenère cipher isn't immune to frequency analysis.

You can break cipher text encrypted with the Vigenère cipher by:

1. Finding the length, `L`, of the keyword that was used for the encryption (What did the video say we need to do to find this length?)
2. Dividing the letters in the text into groups: letters at spots 1, L + 1, 2L + 1, 3L + 1, etc., will go into group 1, letters at spots 2, L + 2, 2L + 2, 3L + 2, etc., will go into group 2, etc.
3. Performing frequency analysis on each of the groups.
4. Substituting the plaintext letters that you found as part of the frequency analysis back into the encrypted text to get the whole plaintext.

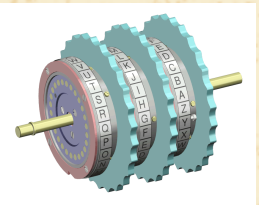
Remember: The Vigenère cipher is just a combination of several Caesar ciphers.

Mechanical Aids: Rotor Machines

A **Rotor Machine** is an electro-mechanical stream cipher device used for encrypting and decrypting secret messages. Such machines were used in the 1920s - 1970s, with the Enigma system being one of its known examples. A rotor machine has rotating disks with an array of electrical contacts on either side. Such a machine implements a fixed substitution of letters, replacing them in a complex fashion.

Specifically, after encrypting each letter, the rotors advance positions, changing the substitution and producing a complex substitution cipher, which changes with every keypress.

During WWII, the Allies broke the encryption of the Enigma machine by analyzing how wires were connected to the machine's rotors and using knowledge about permutations.



Enigma rotor set. Wapcaplet This image was created with Blender., CC BY-SA 3.0, via Wikimedia Commons

Some More Cryptography Terminology

Steganography is the practice of hiding information. For instance, converting a piece of text with wax, writing text with invisible ink, or hiding a message within the pixels of an image.

As you remember from the video on slide 8, steganography isn't an optimal solution to preventing people from accessing data; strong and reliable cryptographic methods should be used instead.

Steganography is an example of **Security through Obscurity**, which is the (usually false) belief that a system is secure as long as its flaws are kept secret. The reason it is usually false is that attackers who are really interested in breaking into a system will, after some time, find and exploit all the previously unknown or secret vulnerabilities.

Cryptology is the practice of encrypting messages and devising new methods thereof.

Cryptanalysis is the opposite of cryptology: it is the practice of 'breaking' messages. One example of cryptanalysis that we already studied is frequency analysis.

Some More Cryptography Terminology

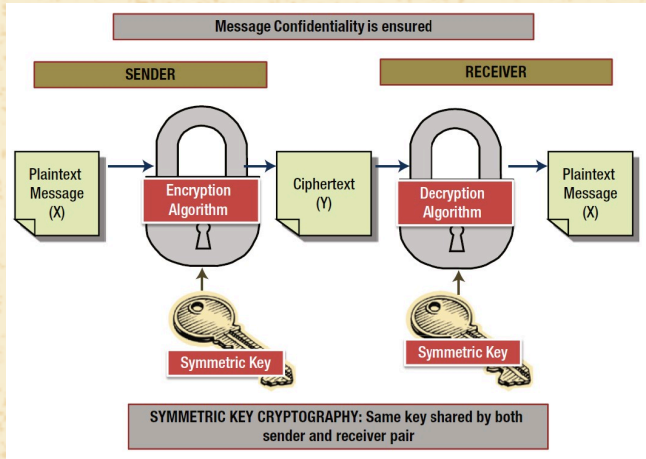
Any questions?

Symmetric Key Cryptography

Symmetric Key Cryptography (also called **Secret Key Cryptography**) is the usage of ciphers that use the same exact key for both encryption and decryption of data/messages.

This key is secret and must be known only to the creator/sender of a message or data and the reader/receiver thereof.

- If anyone else gets the key, they will be able to access the message, thereby violating its confidentiality.



Symmetric Key Cryptography. "Figure 8-3: Symmetric Key Cryptography" (page 166), Nayak, U., & Rao, U. H. (2014). *The InfoSec handbook: An introduction to information security* (1st ed.). APRESS.

Symmetric Key Cryptography

An efficient encryption or decryption algorithm is one that runs in polynomial time. That is, its complexity is `O(n^k)`, where `k` is some constant. That is, linear-time, quadratic-time, cubic-time, etc., algorithms are considered efficient.

A strong encryption algorithm is one that uses randomization when creating the cipher text: each encryption instance may produce a different ciphertext. On the other hand, the decryption algorithm must be deterministic and always output the same plaintext, regardless of the randomization used in the encryption algorithm.

Scenario: Alice and Bob want to send each other confidential messages, but the communication channel is insecure (anyone can listen to the messages going through this channel.) To keep the messages confidential, Alice and Bob could've agreed, ahead of time, on a symmetric encryption system and a secret key K.
If they did so, they can send messages to each other such that no one besides Alice and Bob can understand these messages. For instance, Alice will encrypt a message she wants to send to Bob using the key K, and, once receiving the message, Bob will decrypt it using the same key K. Again, no one else besides Alice and Bob must know the secret key.

Symmetric Key Cryptography

More on Symmetric Key Cryptography:

https://www.youtube.com/watch?v=8Ov4HyncJUU

Symmetric Key Cryptography

An example of a symmetric key cryptosystem is one that uses the **XOR** (= exclusive 'or': $\oplus$) bit-wise operation. Before showing examples of using the XOR operation for both encryption and decryption, let's recall how XOR works.

XOR accepts two bits, Bit 1 and Bit 2 (each of which is either `0` or `1`,) and produces a single bit as a result. The following truth table tells the result of XOR for all the configurations of Bit 1 and Bit 2:

Bit 1	Bit 2	$\oplus$
0	0	0
0	1	1
1	0	1
1	1	0

In other words, if Bit 1 ≠ Bit 2 (`0` and `1`, or `1` and `0`), the result will be `1`. Otherwise, if Bit 1 = Bit 2, the result will be a `0`.

Symmetric Key Cryptography

Example № 1 of using XOR for symmetric encryption and decryption:

- To encrypt the data `1011` using some randomly generated key `1101`, apply the XOR to each pair of bits as follows:

Data	1	0	1	1
Key	1	1	0	1
Ciphertext	0	1	1	0

  because, according to the truth table, we have $1 \oplus 1 = 0$ for the leftmost pair of bits, $0 \oplus 1 = 1$, for the 2nd pair, $1 \oplus 0 = 1$ for the 3rd pair, and $1 \oplus 1 = 0$ for the rightmost pair. The ciphertext is, therefore, `0110`.
- You will use the same key `1101` to decrypt the ciphertext `0110` back to your plaintext data `1011`! Here's why:

Ciphertext	0	1	1	0
Key	1	1	0	1
Data	1	0	1	1

Symmetric Key Cryptography

Example № 2 of using XOR for symmetric encryption and decryption:

- The message "hi", in ASCII binary, is: `01101000 01101001`. We create a random key of the same length: `00111010 10110010`, and encrypt:

Data	0	1	1	0	1	0	0	0		0	1	1	0	1	0	0	1
Key	0	0	1	1	1	0	1	0		1	0	1	1	0	0	1	0
Ciphertext	0	1	0	1	0	0	1	0		1	1	0	1	1	0	1	1

  the ciphertext won't necessarily be English letters. In our case, the ciphertext is: R (R and 'full block'.)
- Again, we use the same key to decrypt the ciphertext back to the message "hi":

Ciphertext	0	1	0	1	0	0	1	0		1	1	0	1	1	0	1	1
Key	0	0	1	1	1	0	1	0		1	0	1	1	0	0	1	0
Data	0	1	1	0	1	0	0	0		0	1	1	0	1	0	0	1

Symmetric Key Cryptography

Conclusions from the two examples:

1. We XORed data with the key to get the ciphertext, and then XORed the ciphertext with the key to get the data. Because the same key was used for both encryption and decryption, XOR is a symmetric cipher.
2. Messages in English can be encoded in (= converted to) the ASCII equivalencies, and then applied with the XOR cipher. That is, we can use the XOR cipher not only on numbers but also on letters, symbols, etc. For messages in other languages, you can use the Unicode table instead.

For the XOR cipher to be considered secure:

1. The size of the key must equal to or larger than the size of the data/message,
2. The key must be generated randomly (such as by using a reliable pseudo-random number generator,)
3. The key mustn't be reused for future encryptions of the same data, and, obviously,
4. The key must be kept secret.

Any questions?

Stream Ciphers vs. Block Ciphers

Symmetric ciphers are divided into two types: stream ciphers and block ciphers.

Stream ciphers are a pair of encryption and decryption algorithms that encrypt/decrypt every bit (or byte) separately. That is, the result of the encryption of one bit won't affect the manner in which the next bit is encrypted.

This type of a cipher is most suitable for the encryption of plaintext that is generated 'on the fly', such as media streams.

Stream ciphers work as follows:

1. A random-like (= pseudo-random) string of bits is generated. This sequence of bits will be our encryption key.
2. Each bit of the plaintext is XORed with a bit from the key.

Notation: `for (i = 0, i < n, i++) ciphertext[i] = plaintext[i] ⊕ key[i];`

Vulnerabilities of Stream Ciphers

Stream ciphers might suffer from several vulnerabilities (weaknesses):

1. **Key Reuse.** In many real-world circumstances, keys have been reused for encrypting messages. This is especially true when the length of a key is small (so the set of all possible keys is small,) leading to an increased key reuse.

   The consequence of key reuse is that if a hacker cracks one of the messages, they will know what all the other messages encrypted with this same key are, without even needing to know what the key is. This is true because:

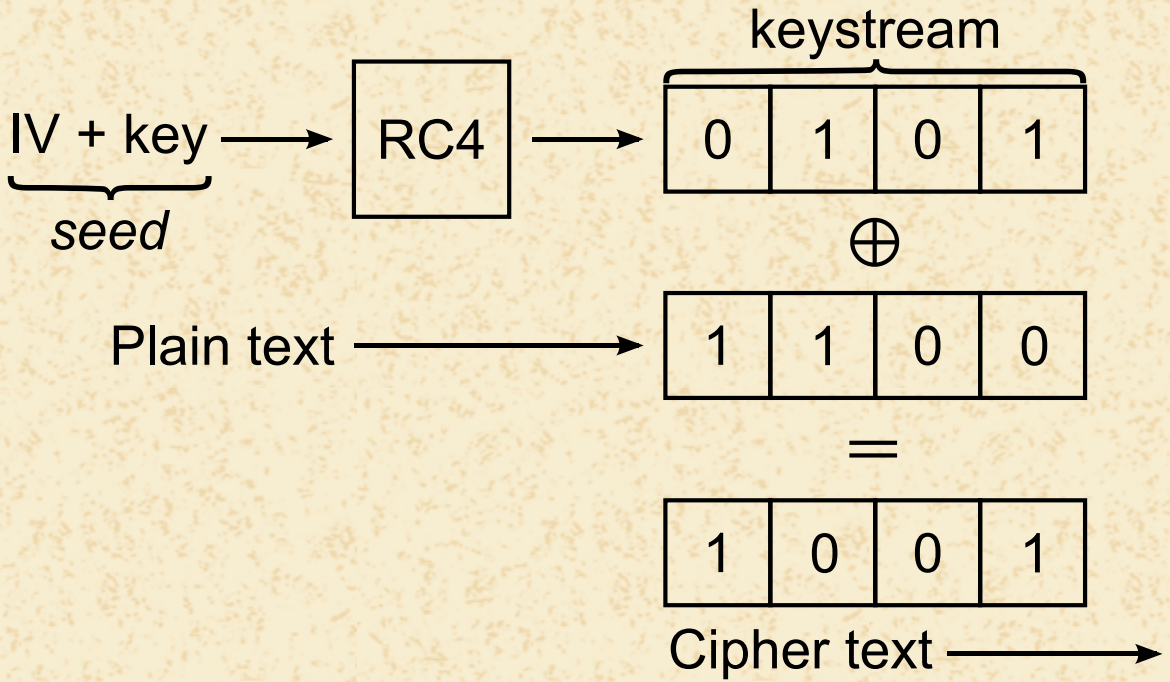
   $$c1 \oplus c2 = (p1 \oplus key) \oplus (p2 \oplus key)$$
   $$= p1 \oplus p2 \oplus (key \oplus key)$$
   $$= p1 \oplus p2$$

   since the key, when XORed with itself, becomes a string of `0`s and so is canceled out. Now, we have $c1 \oplus c2 = p1 \oplus p2$, so: $p2 = (c1 \oplus c2) \oplus p1$, showing that anyone knowing the 1st plaintext message and both cipherstexts will know the 2nd plaintext message!

Vulnerabilities of Stream Ciphers

1. Some known incidents where key reuse led to attacks on the encryption:
   a. During WWI, the U.S. initiated a counterintelligence program named the Venona Project, whose purpose was to decrypt messages transmitted by the intelligence agencies of the Soviet Union.
   U.S. cryptanalysts discovered that the Soviets produced around 35,000 pages of duplicate keys.
   b. The WEP (Wired Equivalent Privacy) protocol, which was intended to provide security to wireless networks (WiFi), is nowadays deemed insecure because of key reuse, too.
      - Each key in this encryption method is built up of two parts: an **Initialization Vector (IV)**, which is a random number, and a key (that never changed.) The digits of the vector were appended to the left of the key (in the form: `VVV...VVKKK...KK`,) and this long number, called the **seed**, was XORed with the plaintext.
      - **The problem** was that the vector is 24-bit in size, so it repeats every 16 million pages (which actually isn't a lot given a computer's speed.)
      - This vulnerability was discovered in 2001, 4 years after the protocol was created.
      - WEP was eventually replaced by WPA2, which is the most secure WiFi encryption solution today.

Vulnerabilities of Stream Ciphers



Basic WEP encryption: RC4 keystream XORed with plaintext. Traced by User:Stannered, original by en:User:Mhandley, CC BY-SA 3.0, via Wikimedia Commons

Vulnerabilities of Stream Ciphers

1. The controls of the key reuse vulnerability are:
   a. Avoid using an initialization vector: simply generate the whole seed using a pseudo-random number generator (so that no part of the seed repeats.)
   b. Never use a key more than once. For instance, whenever a new network connection is created, use a different key that hasn't been used before.
2. **Mallability.** The algorithms of a stream cipher don't guarantee integrity and, therefore, won't be able to detect malicious changes to the ciphertext. An attacker might unnoticeably and still meaningfully change the ciphertext! Example:
   - Eve, an eavesdropper, may suspect that a message from Bob to Alice begins with the words: "FROM: Bob".
   - Eve proceeds to XOR the 9 bytes "FROM: Bob" with the ciphertext, which returns the first 9 bytes of the key.
   - She then uses these 9 key bytes to encrypt: "FROM: Eve" and append it to the ciphertext instead of "FROM: Bob".

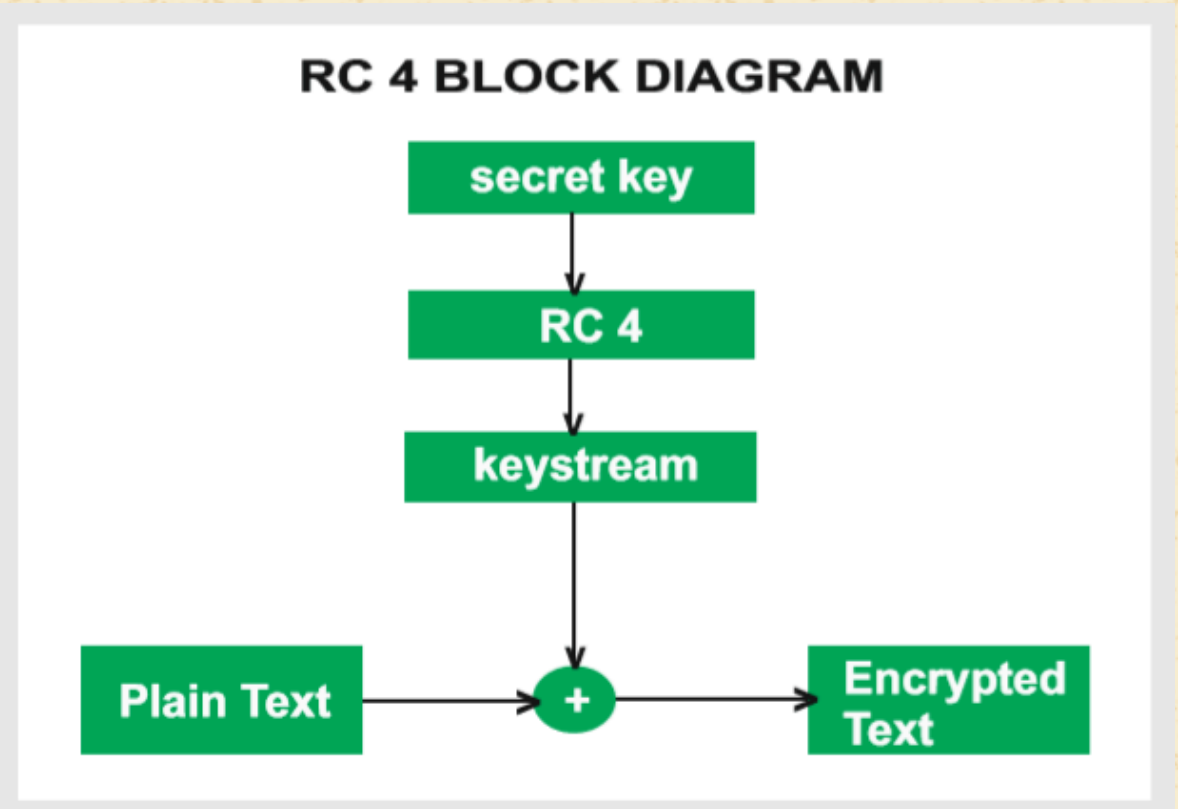
   This was possible because each byte of ciphertext corresponds to one byte of plaintext, so an attacker can target a specific part of a message, without fully knowing the key. This is a breach of integrity!

Stream Ciphers in Practice

Let's introduce two stream ciphers.

1. **RC4** (Rivest Cipher 4) is a symmetric stream cipher algorithm Designed in 1987 by Ron Rivest for RSA Security. Facts about RC4:
   - It uses a seed of variable size: the key can be up to 2,048 bits long.
   - The algorithm takes this seed and generates a pseudo-random stream of bits called the **keystream**.
   - Ciphertext is then obtained by XORing the plaintext with the keysteam.
   - RC4 was widely adopted and used in many popular protocols including WEP, WPA, HTTPS, SSL/TLS due to its simplicity and speed.
   - However, attacks in 2015 on RC4, such as the NO MORE attack, deemed it insecure for use.
   - Many other vulnerabilities were discovered later, such as the increased chance of getting more `0`s than `1`s, making it prone to attacks.
   - RC4 shouldn't be used anymore: any protocol using RC4 is vulnerable to attacks. SSL/TLS ditched RC4 in 2015.

Stream Ciphers in Practice



RC4 Block Diagram. Taken from Geeks for Geeks.

Stream Ciphers in Practice

1. **SSL** (Secure Sockets Layer) and **TLS** (Transport Layer Security) are two security protocols that encrypt the communication between a user (and a user's browser) and a web server.
   Facts about SSL/TLS:
   - TLS is an updated, more secure successor of SSL.
   - Release versions for SSL are 1.0 (never used due to security issues), 2.0 (released in 1995 and deprecated in 2011), and 3.0 (released in 1996 and deprecated in 2015.)
   - Release versions for TLS are 1.0 (released in 1999 and deprecated in 2021), 1.1 (released in 2006 and deprecated in 2021), 1.2 (released in 2008 and is still in use,) and 1.3 (released in 2018 and is still in use.)
   - The letter 'S' in HTTPS stands for 'secure'. HTTPS is the combination of the HTTP protocol, which governs information exchange over the internet, and either the SSL or TLS protocols.
   - All the versions of SSL were deprecated due to security issues, and websites should use TLS for communication encryption.

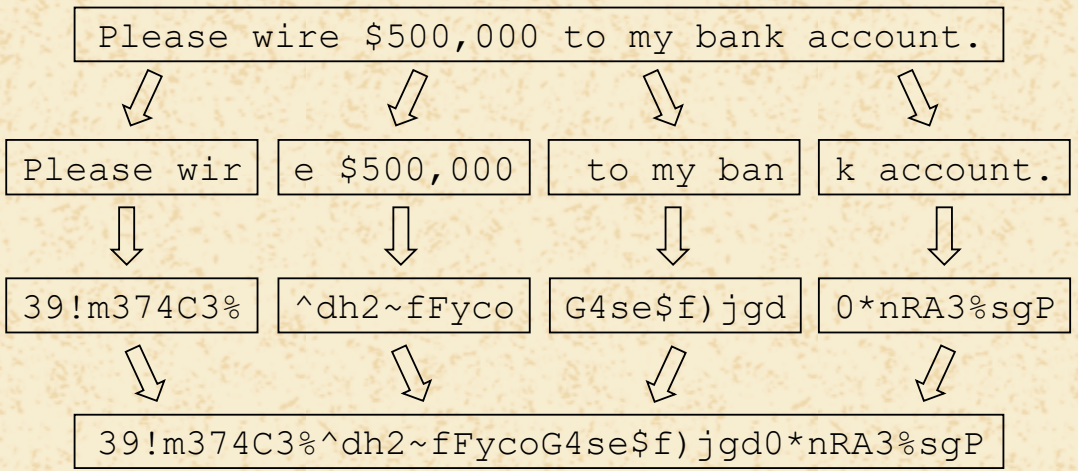
Stream Ciphers in Practice

Any questions?

Stream Ciphers vs. Block Ciphers

The other type of symmetric ciphers is **block ciphers**, which encrypt blocks (chunks) of bits, rather than single bits, at a time.

- All the blocks have the same size: we divide the plaintext into equally-sized chunks and encrypt each chunk.
- This is the most used technique for encryption.



Breaking plaintext into blocks and encrypting them. Miriam Briskman, CC BY-NC 4.0

Block Ciphers: Padding

Sometimes, the length of the plaintext will be a multiple of the block size (as in the example on the previous slide. However, this isn't always the case! This means that the last block will, usually, be smaller than the others. To be able to apply a block cipher, we will pad the last block with `0`s (or another distinguishable character.)

Padding the last block. Miriam Briskman, CC BY-NC 4.0

Block Cipher Modes

Each block cipher consists of an algorithm for encryption and an algorithm for decryption. The **mode** of a block cipher describes the manner in which the algorithms will perform the encryption/decryption of blocks.

Multiple modes were invented:

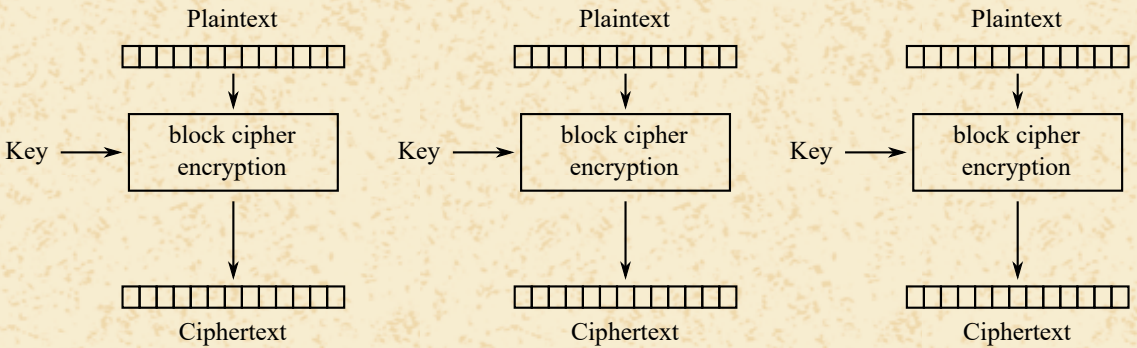
- Providing confidentiality only: ECB, CBC, CTR, etc.
- Provide confidentiality and authentication: GCM, CCM, OCB, EAX, etc.
- Created for special purposes: CMC, EME, etc.

In these lecture notes, we will look into the ECB and CBC modes.

ECB (**Electronic Code Book**) is the simplest mode: each block is encrypted separately by plugging the block plaintext into an encryption function that is given to you. Blocks can be encrypted in parallel (such as by using threads).

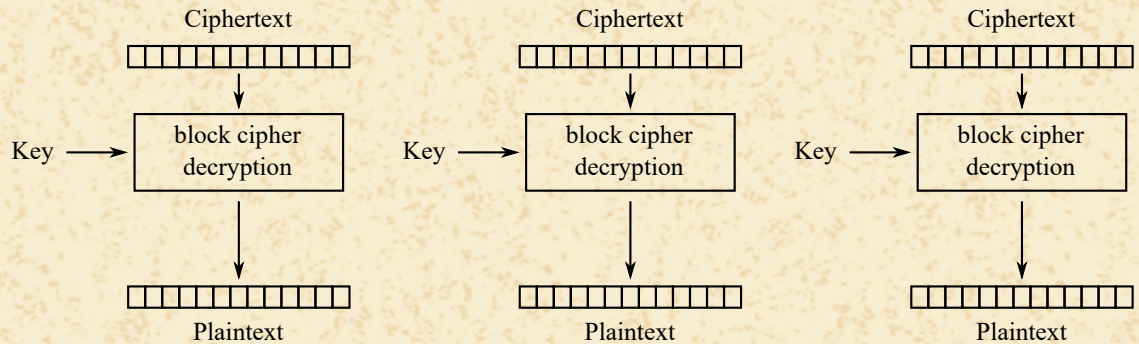
Block Ciphers: ECB

Decryption is performed in a similar way: plug each of the ciphertext blocks into the inverse of the encryption function (a.k.a., the decryption function) to get the plaintext blocks back.



Electronic Codebook (ECB) mode encryption

Encryption using the Electronic Code Block (ECB) mode. WhiteTimberwolf (SVG version), Public domain, via Wikimedia Commons.



Electronic Codebook (ECB) mode decryption

Decryption using the Electronic Codebook (ECB) mode. WhiteTimberwolf (SVG version), Public domain, via Wikimedia Commons

Block Ciphers: ECB

Example of using ECB: Suppose that your plaintext is: `010 110`, which are two blocks of 3 bits each. Also, suppose that you are given the following encryption function $E(b)$ as a table:

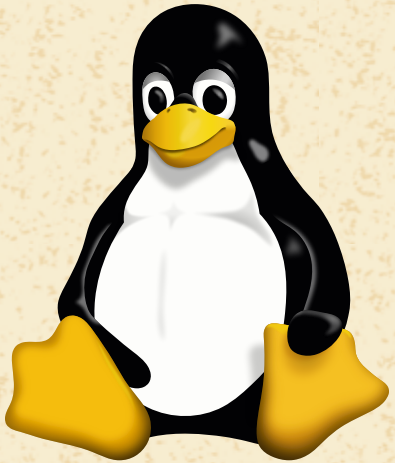
Plaintext	000	001	010	011	100	101	110	111
Ciphertext	101	010	000	001	111	100	011	110

- To encrypt the plaintext, you'll plug each block into the function, separately, to get the ciphertext block.
  - In our case, we do: $E(010) = 000$, and $E(110) = 011$, so our ciphertext is: `000 011`.
- To decrypt the ciphertext, solve for the block variable b by looking back at the table as follows: $E(b) = 000 \Rightarrow b = 010$ and $E(b) = 011 \Rightarrow b = 110$, so we got our plaintext `010 110` back.

Fun Class Activity: Use the encryption table $E(b)$ to (a) encrypt the blocks: `101 100` (b) decrypt the blocks: `001 010`.

Block Ciphers: ECB

The weakness of ECB is that patterns in the plaintext are repeated in the ciphertext, making it easy, if not obvious, to guess what the plaintext is hiding. Other, more secure modes, such as CBC make the plaintext unrecognizable.



Tux, the Linux penguin mascot. lewing@isc.tamu.edu Larry Ewing and The GIMP, CC0, via Wikimedia Commons.

Tux encrypted with ECB: we can clearly see Tux! lewing@isc.tamu.edu Larry Ewing and The GIMP, CC0, via Wikimedia Commons.

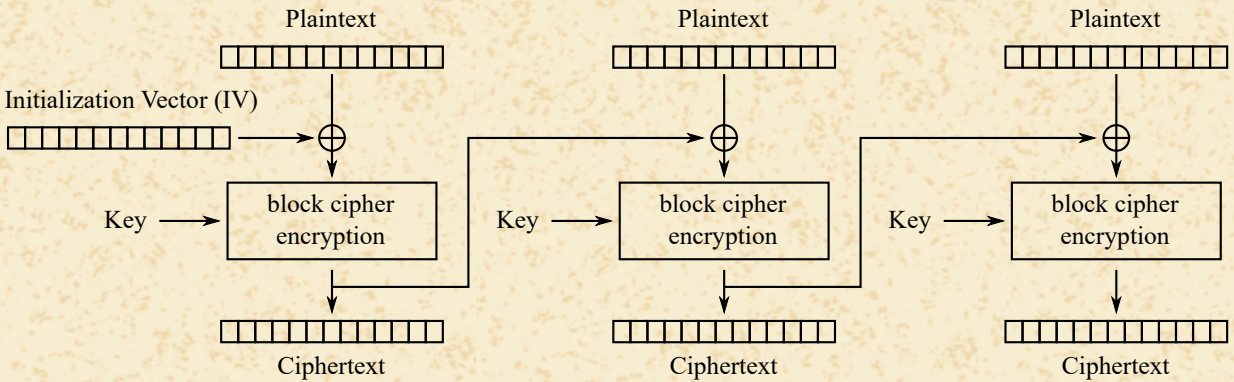
Tux encrypted with a different, more secure mode. Rumpelstilzchen 666, Public domain, via Wikimedia Commons.

Block Ciphers: ECB

Any questions?

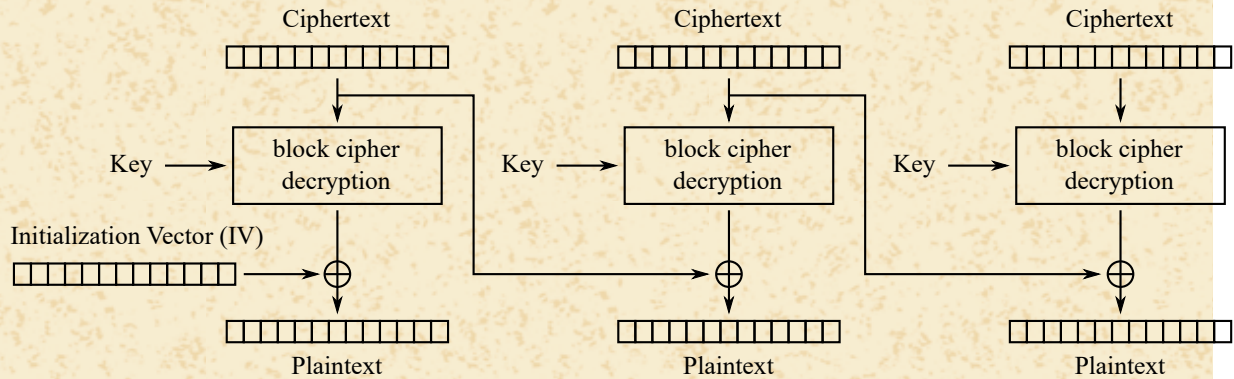
Block Ciphers: CBC

The **CBC** (**Cipher Block Chaining**) mode solves this issue of ECB: it uses the ciphertext of the *current* block to encrypt the *next* block, making blocks depend on one another (a.k.a., "chaining" them) and, thereby, eliminates pattern recognition!



Cipher Block Chaining (CBC) mode encryption

Encryption using the Cipher Block Chaining (CBC) mode. WhiteTimberwolf (SVG version), Public domain, via Wikimedia Commons.



Cipher Block Chaining (CBC) mode decryption

Decryption using the Cipher Block Chaining (CBC) mode. WhiteTimberwolf (SVG version), Public domain, via Wikimedia Commons

Block Ciphers: CBC

Besides making blocks depend on one another, CBC also includes a randomization component: the first block is XORed with a random number/string called the **initialization vector** (also known as **initial value**). In particular, when we encrypt the same plaintext with two different initial values, we get totally different ciphertexts. After we get the result of the XOR, we plug it into the encryption function/table to get the ciphertext of the 1st block.

Then, to encrypt all the subsequent plaintext blocks, we XOR a block with the ciphertext of the previous block and then plug the XOR result into the encryption function/table.

The strengths of CBC are: (1) its relative speed and simplicity and (2) its great ability to hide patterns in text or images. These are two reasons that CBC is the most commonly used encryption mode.

Still, two weaknesses of CBC are that (1) encryption is not parallelized: the blocks must be encrypted sequentially, in a linear manner. Also, (2) CBC only encrypts data and provides confidentiality; it doesn't enforce authentication, so you can't confirm the identity of the sender of that data. Use an additional mechanism, e.g., digital signatures, to provide authentication.

Block Ciphers: CBC

Example of using CBC: Suppose that your plaintext is: `010 110`, and the initial vector is `101`. Also, suppose that you are given the following encryption function $E(b)$ as a table:

Plaintext	000	001	010	011	100	101	110	111
Ciphertext	101	010	000	001	111	100	011	110

- To encrypt the plaintext, you'll work on one block at a time.
  - First, we XOR the 1st plaintext block with the initial value to get: $010 \oplus 101 = 111$, and then plug the result into the table to get the ciphertext: $E(111) = 110$.
  - To get the 2nd ciphertext block, we first XOR the 2nd plaintext block with the previous ciphertext, 110, to get: $110 \oplus 110 = 000$, and then plug the result into the table to get the ciphertext: $E(000) = 101$.
- To decrypt the ciphertext, you perform the actions above in the opposite order: you first solve $E(b_1) = 110$, and XOR b_1 with the initial value to get the 1st plaintext block. Then, you solve $E(b_2) = 101$, and XOR b_2 with the 1st ciphertext block to get the 2nd plaintext block.

Block Ciphers: CBC

Any questions?

Cryptography: Sources

Rao, Umesh Hodeghatta, and Umesha Nayak. "Chapter 8: Cryptography." *The InfoSec handbook: An introduction to information security.* Springer Nature, 2014, pp. 163-181. URL: https://link.springer.com/content/pdf/10.1007/978-1-4302-6383-8.pdf#chapter+8:+cryptography