# AEP 1: Encrypting and decrypting messages with binary

## Overview

In this AEP, you will learn about a way of **encrypting** a message — encoding it so that only the sender and the intended recipient can read it — that uses base 2 representations of integers and a special operation on binary integers known as **XOR**.

**Initial deadline: Sunday, September 24, 11:59pm ET**. Reminder of how AEP deadlines work: This *initial* deadline is the last date by which you are allowed to submit a *first* draft. No *first* drafts are accepted past this date. However, once you submit a first draft on or before this deadline, there are no deadlines for revisions until Sunday, December 3. If your work needs revision, you can continue to revise and resubmit without penalty and without deadlines until that date. The only restrictions are from the course syllabus:

* Only two submissions of AEP work are allowed per week; and
* Your initial draft must be a good-faith effort at a complete and correct solution prior to the initial deadline; otherwise no revision will be allowed.

You are allowed to spend one token to extend this deadline 24 hours.

## Background

We keep secrets from each other all the time. Those secrets aren’t necessarily incriminating or bad; for example, our credit card numbers, social security numbers, and browsing histories are all information that we’d rather not have leaked into the public eye. To keep that information secret in a digital form, we use **encryption**. Encryption refers to *any process that transforms information into a format that is readable only to the owner of that information and the people or machines who the owner decides should see it*.

When encrypted information is sent, the recipient turns it back into a readable format by **decrypting** it. Both encryption and decryption use a **key**, which is a piece of information (a number, a code phrase, etc.) that, like a physical key, is used to lock and unlock the information being sent.

This AEP is about a method of encryption and decryption that was very common in the early days of digital communications, and which (unfortunately, as you’ll see) is still used in some places today. It involves transforming human-readable messages into binary integers, then using a binary string as a key to encrypt it, via a special operation called **XOR**. In this AEP you’ll learn how to encrypt and decrypt using this **XOR cipher**, and play the role of an adversary by breaking an intercepted message encrypted using the XOR cipher.

Here’s a description of how the XOR cipher works.

First, the cipher involves the **XOR** operation. This is a bit operation, denoted xor, that is similar to our logical operators from class. ([In fact you can find more about it in this vault article](https://publish.obsidian.md/mth225/Logic/Exclusive+OR); XOR is shorthand for “exclusive or”, a term that’s explained in the vault article.)

The xor works on single bits as follows:

* 0 xor 0 = 0
* 1 xor 0 = 1
* 0 xor 1 = 1
* 1 xor 1 = 0

Xor is like regular binary addition except for the last line — **we don’t ever carry a 1**. Ordinary addition would say 1 + 1 = 10 but here, 1 xor 1 is just 0. So *xor-ing two bits together always leads to just a single bit*.

We can therefore xor *strings* of bits together, just by xor-ing the individual bits in corresponding positions. For example:

* 10 xor 11 = 01 (because you xor the two bits in the left position and the two in the right position)
* 1011 xor 1111 = 0100
* 10110001 xor 01010101 = 11100100

Now we can describe how the cipher works:

1. Start with a message, in regular text, to send. This is called the **plaintext**. For this AEP, *assume that messages only involve capital English letters — no lower-case letters, punctuation marks, numbers or other symbols*.
2. Take each letter in the plaintext and convert it to a number between **65** representing “A”, and **90** representing “Z”. For example the letter “M” is assigned the number 77. This is the [ASCII representation](https://www.ascii-code.com/) of the character.
3. Then convert each of those integers to their binary representation using an 8-bit format. For example, the letter “M” was the decimal number 77 which in 8-bit binary is 01001101. Put a 0 bit in the leftmost position if needed to get it to 8 bits.
4. Now choose a random 8-bit binary integer to use as your key. *Share this with the person or machine you want to receive your message*.
5. (**Encryption stage**) Your plaintext message is currently a sequence of 8-bit strings. **Take each one, one at a time, and xor it with the key.** Do this for each “letter” in the message. *The result is the encrypted version of the message*. Send it off to your recipient.
6. (**Decryption stage**) When the recipient gets the message, they have the key you sent them (it’s the same key as the one you used to encrypt). The recipient breaks the message into 8-bit blocks. Then, they take each block one at a time and xor the encrypted block with the same key that was used to encrypt.
7. Now take each 8-bit block, convert back to a decimal number, and then convert back into a letter (reversing the process in steps 2 and 3).

If everything is done correctly, the recipient will have the original message in the end.

[!NOTE] **Example:** Alice wants to send Bob the message RUN. First she converts the letters into their ASCII decimals: 82 85 78. Then she converts each of these to binary “blocks”: 01010010 01010101 01001110. Then, she randoly selects the 8-bit string 11000011 to use as the key. (You could use [a random bit generator](https://catonmat.net/tools/generate-random-bits), or flip a coin 8 times, etc.) She shares this key with Bob.

To encrypt, Alice XORs the key with each of the blocks. The first block encrypts to 01010010 xor 11000011 = 10010001. The other two are 10010110 and 10001101. (Double check this to check your understanding.) The secret message Alice sends to Bob is 10010001 10010110 10001101. Notice, a person intercepting this message would just see a string of seemingly-random bits.

To decrypt, Bob takes each block he receives and XORs it with the key (which Alice has shared with him). The first block is 10010001 xor 11000011 = 01010010. Check that the second two end up as 01010101 and 01001110. Bob then converts these to base 10: 82 85 78; and then back to characters using the ASCII tables: R U N.

## Tasks for this AEP

1. Make up a short (8 characters or less) message in English and a key, and encrypt the message using the XOR cipher. Show each step of the encryption process. Make sure to state clearly what your key is.
2. Go to the Google Doc that’s been set up for this AEP: <https://docs.google.com/document/d/1HN5gvs5SvQhwnDsP1adJsUMpWUIKLGhJriJwiWPOkpA/edit?usp=sharing> There, you’ll post your name and your encrypted message (in binary) and your key in a table.
3. One you post your message, look at the entry for the student just above you in the table. **Decrypt the message that they posted, using their key.** Show all your steps here in your writeup. (You’re playing the role of “Bob” in this task; the other person is “Alice”.) If you are the first person to post on the board, you can proceed with the rest of this assignment while you wait for someone else to post.
4. Also found on the Google Doc is a message from Prof. Talbert that’s been encrypted. It’s located at the end of the document in a green box. Pretend you are “Eve”, an evil eavesdropper trying to hack the message. You don’t know the key, because Alice was smart enough not to post it. But you suspect that the binary string 10001111 in the ciphertext was originally the letter E in the plaintext, since E is the most commonly occurring letter in the alphabet and this string 10001111 happens three times, which is more common than any of the other strings. Using this assumption, find the key to the cipher, and decrypt the message (all the way back to English); and then fully explain what you did. Then write a short paragraph that talks about why the XOR cipher can’t really be considered as a “secure” way to encrypt data, based on this task.
5. Think about the complete process of encrypting and decrypting a message, starting from the binary form of the plaintext: We take the 8-bit string that represents the ASCII code for a letter, then XOR it with the key and send it; then XOR again with the same key; and it “magically” comes out to be the original, unencrypted bitstring. **Explain why this process always works;** that is, explain why taking an 8-bit string, XORing with a key, then XORing again with the same key always results in the original bitstring. **Do not just give particular examples that illustrate the idea – give a full explanation for why this process works in general.** You might look at some examples to get you started, but your explanation should be free of specific choices of keys and messages. Use only the properties of binary and the xor operation – and use plenty of English (remember you are giving an *explanation*).

## Expectations and Grading Criteria

AEPs are marked *Success*, *Revise*, or *Incomplete* on the basis of completeness, overall correctness, presentation, and the quality of your explanations. **The primary criterion is the quality of your explanations.** Submissions that are just computations with no explanations attached will be marked *Revise* or *Incomplete* with instructions to provide explanations.

In particular:

* **Your audience for AEP submissions is a classmate – someone who knows all the math you do, but has no familiarity with your solution to these problems**. Your work should be sufficiently complete and clear so that this person can fully understand your reasoning, and agree with it, without having to do any significant extra work. Your submission is graded from this standpoint as well, so if it’s reasonable to think that a normal MTH 225 student would fail to grasp a part of your work without having to fill in gaps themselves, you’ll be asked to revise your solution until this is no longer the case.
* **Most good technical communication is a combination of computations and plain English.** Work that is *all* computation and *no* verbal explanation will likely not meet the standards for “Success”; likewise if the work is *nothing but* verbal explanation.

Please carefully review the [guidelines for AEP submissions in the Standards for Student Work document](https://github.com/RobertTalbert/discretecs/blob/master/MTH225-Fall2023/course-docs/standards-mth225-f23.md#standards-for-aeps) before submitting your first draft, to save yourself time and effort.

In addition to these general requirements, on this assignment “Success” requires the following:

* In Task 4, state clearly what the key is and what the original message was, and fully and clearly explain your thought processes so that an “outsider” could replicate your steps on a new intercepted message. The explanation needs also to use the information provided — if you can break the code some other way, then you can provide that as separate information, but your main solution should use the fact that E encrypted to 10001111.
* Task 5 needs to be a general explanation that explains why the decryption step *always* works. Do not just give additional examples illustrating that decryption works.

## Submitting your work

**AEP submissions must be typewritten and saved as either a PDF or MS Word file. No part of your submission may involve handwriting; work that is submitted that contains handwriting will be marked *Incomplete* and returned without feedback.** This includes electronic handwritten docments, for example using a stylus and a note-taking app. To type up your work, you can use MS Word or Google Docs (both of which have equation editors for mathematical notation) or any other computer-based math typesetting tool. Just make sure you save your work as a Word document or PDF (no .odt, .rtf, or other file extensions are allowed).

If you are writing this assignment up in a Jupyter notebook, you’ll need to share a **link** to your notebook rather than the notebook itself, because Blackboard will not render \*.ipynb files. Just make sure your permissions are set to “Everyone with the link can comment” and then paste the link into the submission area.

When you have your work typed up, double-check it for neatness, correctness, and clarity. Then simply submit your document on Blackboard, in the **AEP** area, in the **AEP 1** assignment.

## Getting Help

You **may** ask me (Talbert) for help on this assignment in the form of **specific mathematical or technical questions, or clarifying questions about the instructions**. If I cannot answer a question because it would give too much away, I’ll tell you so. However please note: **I will not “look over your work” before you submit it to give you feedback on the overall submission**. I have made the expectations clear, so just follow those directions and submit your best work, and you’ll be allowed to revise it if needed.

For AEPs, the syllabus policy on collaboration is:

On *AEPs*, you are **allowed to engage in general discussions of strategy only with others, but no collaboration on the details of a problem are allowed.**.

The safest approach is simply not to discuss any part of your work on an AEP with anyone except the professor.

However: **You can ask technology related questions to anyone at any time**. For example if you need help figuring out how to type up your work, there are no restrictions on that.