AEP 4: Public-key cryptography

> [!WARNING]
> **Use the PDF version of this document! DO NOT use this GitHub/Markdown version**. This version does not render all the mathematical expressions correctly and your work will be wrong if you use it. It is kept on GitHub for archival purposes only.

Overview

This AEP goes back to some of the basic ideas of computer arithmetic to learn about the powerful concept of **public key encryption** --- a method of encrypting and decrypting messages without sharing private information beforehand. The successful implementation of public-key encryption in the 1980's is widely considered one of the greatest scientific innovations of the last few centuries, and we use it every day without realizing it. And it all boils down to arithmetic!

Initial deadline: **Sunday, November 19 at 11:59pm ET**. You are allowed to spend one token to extend this deadline by 48 hours. **However: Please note there is one task below that must be completed by the end of the day on November 16, in order to complete the rest of the assignment**.

Background

This AEP focuses on another method for **encrypting** and **decrypting** messages that has a fundamental difference from the other methods we've seen in class and in AEP 1: Encrypting messages with binary. Before going any farther, please review AEP 1 for an overview of basic terminology and concepts from cryptography such as *encryption*, *decryption*, *plaintext*, *ciphertext*, and *key*.

When we've encountered cryptosystems in the course, *Alice* is trying to send a message to *Bob*, but she doesn't want an evil eavesdropper *Eve* to be able to read it, even if the message is intercepted. So Alice encrypts the message and sends it to Bob. Simple, right? But there's one problem: In all of those earlier systems, **Alice and Bob have to share the key**. It's a lot like if I wanted to grant you access to my office --- I'd need to somehow get you my physical key. But...

- ...What if sharing the key for encryption and decryption was problematic? For example, what I believed that "Eve" might be eavesdropping on the channel I am using to send you the key? I could encrypt the key, but that doesn't solve the problem, because how will I get you the *second* key that encrypts the first one, in a way that convinces me Eve hasn't stolen it or tampered with it?
- ...What if I wanted to send you a message but also wanted to remain anonymous --- like a whistleblower who wants to send an encrypted message to a reporter? Traditional cryptosystems fail on this front because they require that I reveal my identity to you in order to send the message.

What solved both of these problems was a mathematical innovation discovered in the late 1970s and early 1980s called **public key encryption**.

In a public key system, each user creates two keys: one that is kept private, and another that is made 100% public. If Bob is a user, and Alice wants to send him a message, Alice looks up his public key information and uses it along with an algorithm to encrypt the message. Then, when Bob receives the message, he uses his private key to decrypt it. Public key system are made in such a way that the mathematical combination of the public and private keys will decrypt the message.

Another important facet of public-key encryption is that **a person can send an encrypted message to another, without having to make up their own key.** As you'll see, if Alice wants to send Bob a message, *only Bob* needs to have public and private keys; Alice does not need to register with the system.

This AEP will focus on a simple (but vulnerable, as you'll see) public-key system that uses the modulus ("mod") function. Here's how it works (an example of use will follow):

- To generate the keys, first choose four positive integers: a, b, A, and B. These are preferably random, but they can be anything as long as they are positive integers. Make your numbers in the 2-5 digit range; bigger is better but also a little more cumbersome. *The upper- and lower-case of the varaiables is essential*.

- Then do four computations:
  - $M = ab - 1$
  - $e = AM + a$
  - $d = BM + b$
  - $n = \dfrac{ed - 1}{M}$
- The **public key** for the user is the pair of numbers (e, n). The user puts this key next to their name in a public place (a server, Google Doc, skywriting, etc.)
- The **private key** for the user is the number d. The user keeps this private under all circumstances.

If Alice wants to send Bob a message, she does the following:

- Alice looks up Bob's public key (e, n), just like she might search up Bob's email address if she wants to send him an email. (Notice, Alice herself *does not* need a key --- just Bob.)
- Alice takes her plaintext message --- which for this AEP we will assume is just a string of capital letters without punctuation or other symbols --- removes all spaces, and converts each letter to numbers 00 through 25 according to its position in the alphabet, and puts them all together as one long string of decimal integers (possibly with one or two leading zeroes).
- Alice then breaks that string up into equal-sized blocks, each of which is an integer less than n (from Bob's public key), and adds zeroes to the end of the final block if needed to make it the same length as the others.

Now to **encrypt** this message, Alice goes through the message one block at a time. For each block P, Alice computes: $C = (e \cdot P) \,\%\, n$. Once each block has been encrypted, the chain of blocks is the ciphertext that is sent to Bob.

To **decrypt** the message, when Bob receives the ciphertext, which is a chain of blocks of integers, for each encrypted block C, he computes: $(C \cdot d) \,\%\, n$. This produces the decrypted blocks that Alice sent. (*But why?*) Bob can then separate the blocks into two-digit units and convert back to letters, and read the message.

Example

In order for Alice to send Bob a message, Bob (not Alice) first needs a key. Bob randomly generates the numbers $a = 86$, $b = 62$, $A = 26$, and $B = 34$. Then he computes:

- $M = ab - 1 = (86)(62) - 1 = 5331$
- $e = AM + a = (26)(5331) + 86 = 138692$
- $d = BM + b = (34)(5331) + 62 = 181316$
- $n = \dfrac{ed - 1}{M} = \frac{(138692)(181316) - 1}{5331} = 4717141$

The pair $(e = 138692, n = 4717141)$ is the public key; Bob decides to get it put onto a T-shirt that he wears when he's going out, because who cares if it's public? If this system works as advertised it shouldn't matter if people know his public key. But, the number 181316 is private, so Bob tells nobody what it is, under any circumstance!

Alice wants to send Bob the message `AM ON MY WAY`. Converting this to integers 00-25 gives `00 12 14 13 12 24 22 00 24` and stripping out the spaces gives `001214131224220024`. Bob's value of n is 4717141. So Alice breaks her message up into blocks less than 4717141 --- for example, into blocks four digits long: `0012 1413 1224 2200 24`. To get that last block to be exactly four digits, she pads it with a couple of zeroes to get `0012 1413 1224 2200 2400`. (She could also have chosen blocks five digits long, or shorter.)

Now Alice encrypts each block as if it were a four-digit integer. For example, the second block in the plaintext is `1413`. To encrypt it, Alice takes Bob's public key --- which is the pair $e = 138692$ and $n = 4717141$ --- computes

$$(e \cdot P) \,\%\, n = (138692 \cdot 1413) \,\%\, 4717141 = 195971796 \,\%\, 4717141 = 2569015$$

And that's the encrypted version of that block. Repeat this computation for each of the other blocks. The collection of encrypted blocks is then sent to Bob.

Bob receives Alice's incoming encrypted message, which looks like a bunch of integers, which could be up to seven digits long. The second one in the sequence is 2569015. This represnts two encrypted text characters. To decrypt, Bob pulls up his private key d along with n from his public key and computes:
$(C \cdot d) \,\%\, n = (2569015 \cdot 181316) \,\%\, 4717141 = 465803523740 \,\%\, 4717141 = 1413$
Bob can then tell that these are the characters `O` (14) and `N` (13). Bob can continue doing this with each encrypted block until he has Alice's original message.

Notice that Bob's private key was never communicated; and Alice needed no key of her own to send the message. The only thing Alice needed was Bob's *public* key pair, which the whole world has access to.

Tasks for this AEP

1. Generate a set of keys for yourself using the method above method. Keep the private key d completely private, but take your public key pair (e, n) and put it on this spreadsheet along with your name: https://docs.google.com/spreadsheets/d/1CO4W3wmT8fE_WrCHlJ9SIQv8HWNQHqJ8S8bL8Lv0_gw/edit?usp=sharing **IMPORTANT: This step must be completed no later than Thursday, November 16 in order to be able to complete Task 3 below and submit a complete draft on time!** Without your public key I cannot send you an encrypted message for Task 3. If your public key info is not posted by the end of the day on 11/16 it means you do not intend to complete the assignment.
   Don't put any more details on the spreadsheet. In your writeup, show all the steps you used to generate the keys. (You can do this with Python code or with a calculator.)
2. Make up a short message, encrypt it using my public key, post it with your name on this Google Doc: https://docs.google.com/document/d/1IXaVRMFEYx-cI8KgBQAUP-4YV5Z3zLoy7tyOt1ELfKA/edit?usp=sharing. In your writeup, state your unencrypted message and then show all the steps you used to encrypt it and arrive at the encrypted message. Remember since you are sending *me* the message, you have to use *my* public key --- not yours.
3. Once you have posted your public key, I will send *you* an encrypted message by email. It will be something where it's obvious if you've decrypted it correctly. When you get it, put all your math work used to decrypt the message in your writeup.
4. For the final task, you have two options.
   - *Option 1: Hack the professor*. As you saw, my public key information is *public*, there for the world to see on an open spreadsheet. In a secure system, a hacker should find it difficult or impossible to figure out my private key even if my public key is exposed. However, it turns out this system has some holes in it. **Figure out what my private key, d, is --- using only my public key, and knowledge of the formulas used to generate the keys.** Obviously in your writeup, you'll need to state my private key and then *clearly explain* how you got it, in complete detail.
   - *Option 2: Explain how all this works*. Give a clear, correct mathematical explanation for two parts of this process that aren't totally obvious: First, explain why $n = \frac{ed-1}{M}$, although defined as a fraction, will always work out to be an integer; and second, explain why for any positive integer x, if we perform the math operations to encrypt x, and then perform the math operations to decrypt the result of the encryption step, we will end up with x each time. (In other words, explain why this system will always "work" in the sense that the decryption of an encrypted message is the plaintext.) For this option, the explanations will involve quite a bit of algebra and computation --- but be sure to add English to help the reader understand.

Expectations and Grading Criteria

when it comes to **explanations**:

- **Keep it simple and clear.** Do not assume that the reader is a genius, has lots of experience shared with you, or wants to fill in any gaps you leave. Use plain language and make your explanations as clear as possible.

- **General statements are not explained by examples.** For example if you think a function is injective, it means that there are no collisions from the input to the output. Explaining this, requires that you explain why collisions **never** happen with this function; a single, specific example of a collision not happening isn't convincing. However, remember that if a function **is not** injective, or is not surjective, then this *can* be explained with a single example where the property fails, so you can give that example and explain why it works.
- **Remember the audience**. The audience is **your classmates** --- people who are smart and have all the background knowledge that a regular person in the class would have, but none of the specialized knowledge that *you* have, and no experience with your particular problem. At regular intervals, stop and ask yourself if, honestly, your explanation would be clear and convincing for that audience. If not, keep working on it.

And remember: **The main criteria used for evaluating your work on this and most other AEPs is the *quality of your explanations, not the correctness of your answers***. It's pretty easy to get right answers; but it takes real understanding to explain why they are right.

Remember the general expectations and grading standards for AEPs are in the Standards for Student Work document. Make sure to review this before submitting anything and use those guidelines as a checklist to save yourself time and energy later from having to revise work that doesn't meet those standards.

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 graded N 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).

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 4** assignment.

There are no special additional criteria for this AEP. Although there is code to analyze in this AEP, you do not need to *write* any code for it, so a Jupyter notebook isn't necessary. If you choose to use a Jupyter notebook anyway, that's OK, just remember to set permissions on your document correctly ("Everyone with the link can comment") and turn in the link, not a PDF.

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..

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.