Cryptography Basics

To achieve secure communication, we *encrypt* any plain information (called *plaintext*) into a corresponding *ciphertext,* using some critical information (called *key*). We then send the ciphertext to our trusted parties, who will *decrypt* it back into the plaintext.

In discrete math terms, the key actually operates as an *invertible function*:

$$\begin{aligned} f(\text{plaintext}) &= \text{ciphertext} \\ f^{-1}(\text{ciphertext}) &= \text{plaintext} \end{aligned}$$

A very crude way of categorizing crypto systems is the following: they are either *symmetric* or *asymmetric*.

- In *symmetric* crypto systems, there is only one *key* f. Given f, it is very easy to compute/obtain f^{-1}, and vice versa. The oldest crypto systems (before 1970s) are all symmetric.
  - The weakness of such a system is obvious: once f is known to malicious parties, there is no security anymore.
  - f itself therefore needs to be kept as a secret. Then how do we communicate f itself to our trusted parties?
- In *asymmetric* crypto systems, there is a public key f and a private key f^{-1}, such that given f, it is very hard (unreasonably hard) to directly compute/obtain f^{-1}.
  - Therefore, f can be public information. It doesn't hurt if malicious parties know f - they won't be able to find f^{-1}.
  - Only the receiver needs to know f^{-1}; anyone who wishes to send information only needs to know f. Therefore, f^{-1} never needs to be communicated.
  - This is very nice, but the catch is all in the *unreasonably hard* phrase. What is unreasonably hard to compute 20 years ago might not be so today. What is unreasonably hard to compute now might not be so in 2030.

You have all used asymmetric crypto systems.

If you have used **coursework.cs.duke.edu (https://coursework.cs.duke.edu/)** , **gitlab.cs.duke.edu (https://gitlab.cs.duke.edu/)** , or Github, you have probably **configurated your *SSH key* (https://coursework.cs.duke.edu/-/profile/keys)** . That's an asymmetric crypto system. When you "configure your SSH key", your device actually generates a pair of public and private keys. Depending on how you set it up, you might have used the **Ed25519 (https://ed25519.cr.yp.to/)** algorithm, the **ECDSA (https://cryptobook.nakov.com/digital-signatures/ecdsa-sign-verify-messages)** algorithm, or the RSA algorithm. This EM will only discuss the

RSA algorithm, in its most simple/abstract form (which is actually not the form in use in real platforms). This is partly because it is one of the first developed asymmetric crypto systems, but also partly because the math behind it (for its core part) is very accessible.