# Asymmetric cryptography and algorithms on large numbers

We spent a majority of the semester looking at symmetric cryptography. This is fitting because most cryptography is performed this way. But, symmetric cryptography requires shared keys between all communicating parties and this is logistically difficult.

Asymmetric cryptography allows each party to generate and share their own public key and then this public key can be leveraged for secure communication. Because individuals create and publicly share their own key, logistics are simplified.

Algorithms for asymmetric cryptography are dependent on math problems that are easy to compute but hard to invert. The first topic of this module will be learning about these hard problems and how to implement them using algorithms for very large numbers.

We will then look at how two hard problems -- RSA and discrete logarithm -- can be used to achieve cryptographic goals.

**Learning objectives**

By the end of this module you should be able to...

- Simulate algorithms for RSA encryption, decryption, and signing; O(log n) exponentiation via squaring-and-multiplying;  O(log n) multiplicative inverse finding; and random probable-prime generation;
- Explain why "textbook" RSA is unsuitable for secure use and how to remedy it;
- Compute group operations on prime multiplicative groups and additive elliptic-curve groups;
- Simulate Diffie-Hellman key-exchange using a prime multiplicative group or an additive elliptic-curve group;
- Find an efficient subgroup of a prime multiplicative group and explain the benefit of doing so; and
- Explain man-in-the-middle attack in the context of Diffie-Hellman key exchange.