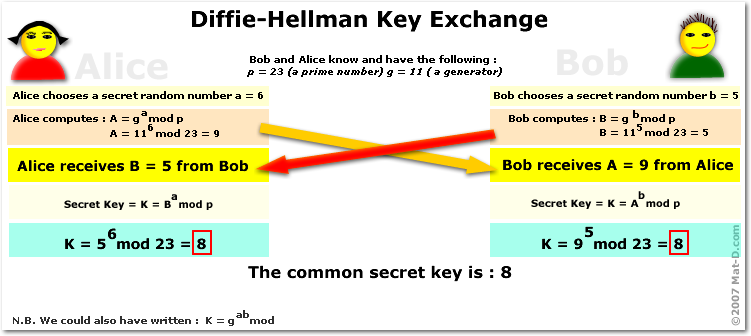
Diffie Hellman Key Exchange

Diffie-Hellman Key Exchange is an algorithm that establishes a shared secret between two nodes using only public data transmission. The algorithm was the first of its kind to solve the fundamental problem of generating a shared key. Previously, keys were often physically transported on paper to ensure security. Following the algorithm, systems could generate secret keys quicker, easier, and more often. Today, it is standardly used to generated private keys for use in symmetric encryption applications.

**Algorithm Overview**

To generate a shared secret number Bob and Alice, our two lovely cryptographic communicators, must follow the following process:

1. Bob and Alice establish and share a modulus, ***p***, and base, ***g***. (modulus must be a prime number and the base is a number smaller than the chosen modulus)
2. Bob and Alice both choose a secret number ***a*** and ***b***, respectively.
3. Using the modulus, base, and the secret number, Bob and Alice both generate a unique number, ***A*** and ***B***:
4. Alice sends Bob ***A***. Bob sends Alice ***B***.
5. Alice and Bob can both generate the shared secret using the modulus ***p***, their secret number ***a***/***b***, and the other’s unique number ***A***/***B.***



***Figure: Diffie-Hellman Example***

**Lab Exercise:**

In this lab we will implement the Diffie Hellman Key Exchange over a CAN bus using two separate Teensy’s each running a separate Bob and Alice sketch. To complete the lab, you will need two functioning teensy loggers and the included Bob and Alice sketches. The sketches use Rhys Weatherley’s arduinolibs to produce random numbers. Download and install the libraries from:

[https://github.com/rweather/arduinolibs](https://github.com/rweather/arduinolibs%20)

While a solution is included, please work through the unfinished code before looking at the solution. As always, there is more than one way to implement this in code and you may find a more efficient implementation. Basically, Alice will generate ***p*** and ***g***, send them to Bob along with her ***A***. Bob in turn responds to Alice with his ***B***. With the exchanged information, both can generate and output a shared secret. The code is structured and all math operations are implemented. Your job is to send and receive the CAN messages exchanging the information. As always, if any questions arise, ask!

**NOTE**: In general, it is typically a poor security decision to implement cryptographic algorithms yourself. Without deep hardware understanding and peer review, the code can lead to vulnerabilities because of how it is compiled and run. This is for demonstration and learning purposes only.