**Message Integrity**

In the world of secured communications, Message **Integrity** describes the concept of ensuring that **data has not been modified in transit**. This is typically accomplished with the use of a Hashing algorithm.

The basic premise is a sender wishes to send a message to a receiver, and wishes for the integrity of their message to be guaranteed.  The sender will calculate a hash on the message, and include the digest with the message.

On the other side, the receiver will independently calculate the hash on *just* the message, and compare  the resulting digest with the digest which was sent with the message.  If they are the same, then the message must have been the same as when it was originally sent.



Pretty straight forward.  Except for one major problem.  Can you guess what it is?

If someone intercepted the message, changed it, and recalculated the digest before sending it along its way, the receiver’s hash calculation would also match the modified message.  Preventing the receiver from knowing the message was modified in transit!

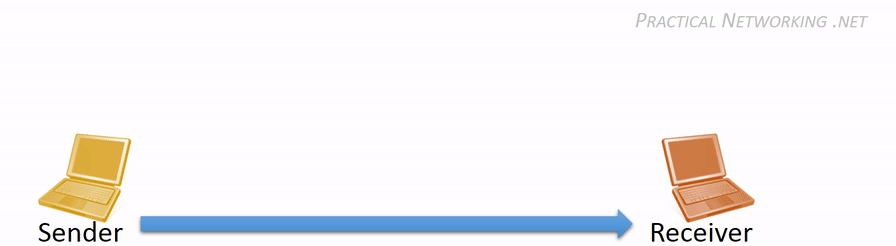
So how is this issue averted? By adding a Secret Key known only by the Sender and Receiver to the message before calculating the digest. In this context, the Secret Key can be any series of characters or numbers which are only known by the two parties in the conversation.

Before sending the message, the Sender combines the Message with a Secret key, and calculates the hash.  The resulting digest and the message are then sent across the wire (*without the Secret!*).

The Receiver, also having the same Secret Key, receives the message, adds the Secret Key, and then re-calculates the hash.  If the resulting digest matches the one sent with the message, then the Receiver knows two things:

1. The message was definitely not altered in transit.
2. The message was definitely sent by someone who had the Secret Key — ideally only the intended sender.

This animation reflects this process:



When using a Secret Key in conjunction with a message to attain Message Integrity, the resulting digest is known as the **Message Authentication Code**, or**MAC**.  There are many different methods for creating a MAC, each combining the secret key with the message in different ways. The most prevalent MAC in use today, and the one worth calling out specifically, is known as an **HMAC**, or Hash-based Message Authentication Code.

Of course, this doesn’t answer the question of “How did the Sender and Receiver establish mutual secret keys?”  This is known as the Key Exchange problem, which comes up a few times in cryptography. However, the answer lies outside the scope of the concept of Integrity, and will be discussed in another article in this series.