Drew Applegath

EE4723

04/24/2017

Implementation of a Secure I2C Communication

The scope of what I did was to implement a shared key version of an I2C communication protocol. Using a light weight version of RC4, the secure I2C protocol that I have implemented can be used to encrypt communications from a controller to any number of I2C devices that it is controlling. Though it is unlikely that anybody would try to tap into an I2C bus to control sensors, this could help validate data in the case that a master is overtaken. This could even be implemented in vehicles or control systems running I2C to ensure that the signals being sent for control are valid. The entire JSON is encrypted using the RC4 library and sent over the I2C bus. Since the I2C protocol does not specify the size for a packet, the strings can be as large as needed for what every form of data needs to be sent.

In the version submitted, there is a location out front of a valid string that contains the key, but when implemented in a working communication, this would be replaced with another form of validation, using a certain key or nonce. The following string would hopefully contain a JSON string of data that can be interpreted by the receiver. The protocol using RC4 is able to work both ways, due to the shared key that is contained.

The secure I2C project contains very basic encryption, and has a very light weight encryption function, but it does the job of scrambling and securing the data being sent from the controller to the device it is controlling. This could be used in many imbedded applications due to its light weight and the relatively light hardware requirements for encrypting data. The only short fall is that the key is hardcoded into each unit, making it very easy to get around if you can determine the key, and there is

no way currently to change the key, because the hardware and time requirements for a public key type system would not be ideal for an embedded system.