## Introduction

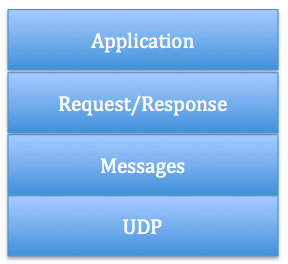
IOTsec is a simple and secure communication protocol with low overhead implemented to satisfy the requirements specified in the case-study.

This protocol is lightweight and efficient to run on devices with limited computational resources and memory. It is suitable for devices with Limited memory and non-volatile storage & minimum computing power (100s of KBs & tens of MHz), battery powered, low bandwidth (Kbps — Kilob*its* per second).

This protocol has been implemented in C, and it can be integrated easily into any C/C++ based application. The end application has to the linked to the IOTsec library to avail the protocol.

## Design

IOTsec is like HTTP's client/server model. Below figure shows that it employs a two-layer structure. The bottom layer is Message layer that has been designed to deal with UDP and asynchronous switching. The request/response layer concerns communication method and deals with request/response message.



## Endpoints Communication

As shown in the previous picture, there are two different layers that make IOTsec protocol: Messages and Request/Response. The Messages layer deals with UDP and with asynchronous messages. The Request/Response layer manages request/response interaction based on request/response messages.

IOTSec protocol messages layer deals with UDP exchanging messages between endpoints. Each message has a unique message ID; this is useful to detect message duplicates.

Each message is built by these parts:

* A binary header
* A compact option
* Payload

When exchanging messages between two endpoints, these messages can be reliable. A message is sent again and again until the other party sends an acknowledge message (ACK).

Below are the points that illustrate the protocol.

* It uses UDP protocol and a compact message format to save on payload size.
* IOTsec follows a client/server messaging model just like the broader internet and HTTP. This makes it quite easy to understand how to send messages to and from say a sensor and a database.
* Devices participating in the communication can be client, server, or both. For example, a single sensor node might be a client publishing sensor data to a cloud server while the cloud is a client watching the state of a device that went to sleep, acting as a server when it wakes up. It can even extend to the local network — two devices can communicate directly acting as client/server pairs.

## Encryption

IOTsec protocol by default uses UDP so it uses DTLS (TLS for Datagrams using GNU TLS) to secure the transport layer. With DTLS as a basis, it defines two security modes:

1. *Certificates:* Certs are what one would consider “enterprise-grade security.” Like RPK this mode uses asymmetric key pairs.
2. *No Security:* Technically not a security mode, and unfortunately the most common mode. This mode is activated by disabling the compilation switch: IOTSEC\_DTLS\_EN

## Integration

It can be easily integrated into any application to establish communication between two endpoints. The Makefile presented in the test application is helpful to understand on integrating with the end application.

## Compilation

Execute the compilation script present in the sources. Execute the below steps to compile the library and test application. This script also produces the rich Doxygen documentation.

$ chmod +x compile.sh

$ compile.sh

## Documentation

Codebase has been extensively commented with Doxygen style. Documentation is generated by running the compiled script as described in the previous section. Doxygen based documentation is available in the doc directory.

## Code footprint

The code footprint of the library is exceedingly small, and it can be fit into the non-volatile memory of the basic STM32 STM32MP157C microcontroller. The functionality has been tested executing the basic use cases on an IOT device with STM32MP157C microcontroller.