# Run Your Own Secure IoT Cloud Server for $8 a Year

This article provides an introduction on how to setup a secure Internet of Things (IoT) cloud server, where memory-constrained edge nodes can communicate securely with the cloud server and where devices can be managed in real time using a web-based user interface. This article is tailored for learning purposes and DIY projects. At the end of the article, we provide a link to the details (the recipe with $8 ingredients) on how you can setup your own secure IoT server and device infrastructure.

## IoT Cloud Server Solutions

Most IoT cloud server solutions, whether they provide ready-to-use hosted services or not, are based on a standard Virtual Private Server (VPS). Most developers probably think about Amazon or Microsoft Azure's services when considering the server side of their IoT solution. These high-end services are great if you need to scale up to millions of connected devices. However, for most small-scale operations and DIY projects, a low-cost VPS is more than adequate.

The website lowendbox.com provides reviews for low-cost Virtual Private Servers and is a great place to start when selecting a VPS. We found an $8/year VPS suitable for our secure IoT experiment. With this VPS, we were able to connect up to 10,000 devices. Connecting more devices requires more memory, but the VPS we selected has memory limitations (64MB). Thus, if your IoT solution requires more connected devices, a VPS with more memory will be required.

VPS providers, including Amazon, typically provide a web-based user interface from where you can manage your VPS. The web-based control panel provides services such as selecting and installing/re-installing the operating system. The operating system of choice is typically a flavor of Linux. After installing the Linux operating system using the web interface, the server will be online and be ready for you to logon and manage.

## Cloud Server Operating System and Software

A freshly installed Linux operating system on an online VPS is typically bare-bone with few services running. At a minimum, the Linux installation must have an SSH (Secure Shell) server running so you can remotely logon to the server and install software of your choice. The software we selected for the server-side IoT solution is an application server called the Mako Server. One of the reasons for selecting the Mako Server is that it uses memory very efficiently. In contrast, most high-end server side application frameworks will require large amounts of memory, and are therefore unable to operate on a low-cost VPS.

The Mako Server is an extremely light-weight application server that can easily operate within the memory limitations of a low-cost VPS. Another reason for selecting the Mako Server is that the server can act as a dual-certificate server, thereby enabling the use standard RSA certificates for browsers and small ECC certificates for edge nodes. SSL (Secure Sockets Layer) certificates can have a great impact on memory if you do not consider the type of certificate being used in your IoT solutions. Using the wrong type of certificate may break the design of a memory-constrained edge node. We will consider this in more detail in the security section below.

## IoT protocol for Secure Edge Node to Cloud Communication

The Mako Server includes a secure IoT protocol called SMQ (Simple Message Queues). The server and its application framework enable the user to write server-side scripts for interacting directly with the SMQ IoT broker. The server-side application framework also enables the user to extend the IoT protocol and connect it to other services on the Internet or to other local services running on the VPS, such as database services.

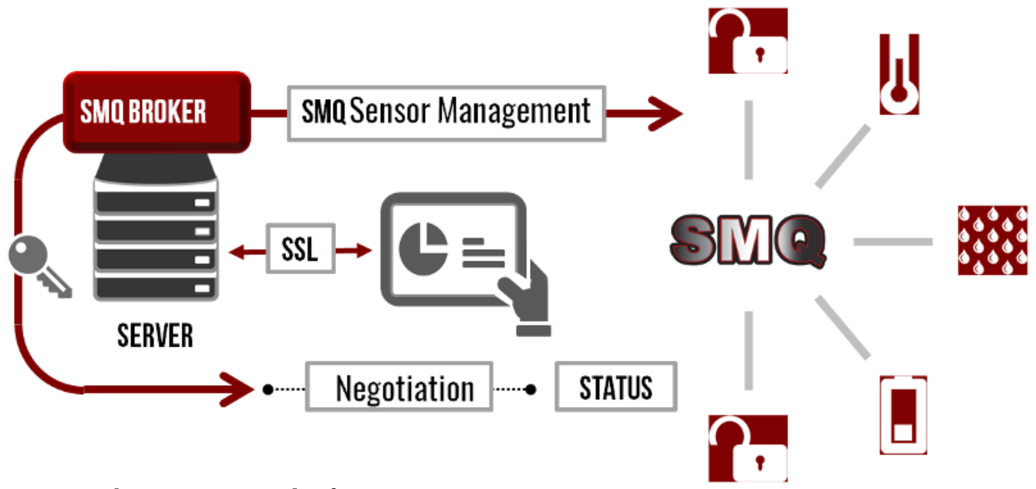

Figure 1, the SMQ protocol infrastructure

The SMQ protocol is ideal when secure real-time data exchange between a web interface and an edge node is required. The protocol, which is similar to the pub/sub mechanism in MQTT, enables web-based user interfaces to control edge nodes in real time. All communication goes via the cloud server, which means you will not run into any network address translation (NAT) traversal issues when communicating between different (local) networks.

## Enabling Trust and Security

Not surprisingly, setting up a secure IoT solution requires more work than setting up a non-secure solution. A secure IoT implementation requires that, at a minimum, you setup a trusted server. The SSL/TLS (Secure Sockets Layer / Transport Layer Security) protocol is used for the encrypted communication, but TLS will not be secure unless the infrastructure is based on trusted X.509 (SSL) certificates. This trust is the key component required for TLS to be secure. For this reason, it is important to install a certificate in the server that is trusted by all of the client's connected to the server.

A browser that connects to the server requires an installed certificate that is signed by a well-known certificate authority (CA). In this case, "well-known" means that the CA's public root certificate is pre-installed in the browser/computer that you are using. You can use free or paid-for well-known CA services when signing your server certificate.

A device that connects to the online server can keep a copy of the CA's public root certificate in the device. This certificate can be extracted from your browser/computer and copied to your device. The device will not be able to validate (trust) the server's certificate if the device does not keep a copy of the CA's public root certificate. The CA's public root certificate is typically embedded in the device firmware.

An alternative to using a well-known CA is for you to become your own CA and to use your own server certificate for device communication. Since you are in full control of the device and can store any type of certificate in the device, including your own public root certificate, you can easily use your own signed server certificate for device communication. Being your own CA is not difficult when you have easy-to-use tools.

One of the benefits of being your own CA for the certificate exchanged between the server and the device clients is that you can select to use an Elliptic Curve Cryptography (ECC) certificate for the server. The benefit with using an ECC certificate is that the certificate is much smaller than an RSA certificate, and thus consumes much less memory in the device during the initial SSL handshake.

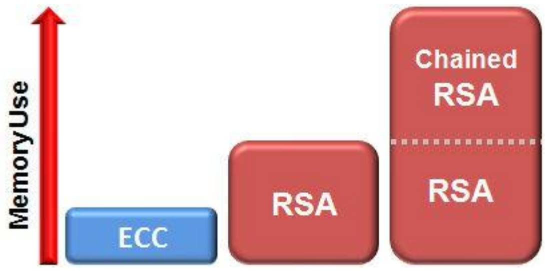

Figure 2 illustrates the size difference between an ECC Certificate, an RSA Certificate, and a chained RSA Certificate

SSL certificates can have a huge impact on memory in constrained edge nodes, thus using a non-chained ECC certificate may be a requirement for a memory constrained device. Most well known CA services only sign RSA certificates and the free/low cost CA providers typically sign certificates with an intermediate CA certificate that requires a chain of trust, thus consuming even more memory in the device.

As was previously noted, one of the benefits with the Mako Server is that it can act as a dual-certificate server, whereby browsers are served a larger (chained) RSA certificate and devices are served a smaller (non-chained) ECC certificate. A dual-certificate server provides great flexibility as it allows the use of small certificates for devices and RSA certificates for browsers. The RSA certificate can be signed by a well-known CA, while the ECC certificate is preferably signed by your own CA.

## Getting Started

If you want to learn more, head over to the Mako Server's IoT page. The IoT page includes a section with instructions for how to setup your own secure server and how to securely connect ARM mbed boards (edge-node devices) to your own cloud server.

https://makoserver.net/smq-broker/