Introduction

The MAXREFDES9001 is a complete Internet-of-Things (IoT) security reference design featuring a LoRa radio based, low-power, temperature sensor node secured with a DS28S60 secure co-processor, a LoRa gateway, and a Google Cloud application. This reference design showcases a robust and easy to manage end-to-end security scheme with authentication and confidentiality capabilities independent of the transmission link in use -- the LoRaWAN protocol in this case. The MAXREFDES9001 is designed to easily integrate into embedded systems enabling confidentiality, authentication, and integrity of information.

The sensor node is motioned by the tiny, low-power, Cortex-M4 based microcontroller MAX32660 which periodically measures the ambient temperature with the help of the DS7505, authenticates and encrypts the temperature value using AES-GCM with the DS28S60 secure coprocessor, and sends it to the Google Cloud application over a LoRaWAN network, via a Raspberry Pi powered gateway. To prevent rogue nodes from publishing data, joining the nodes to the network requires a prior local verification using a convenient NFC based strong authentication with help of the MAX66242 Secure Authenticator and a dedicated Android application running on an NFC enabled Android device. Once this strong authentication is successful, proving that the node device is genuine, the Android device communicates with the Google Cloud application via Internet to provision the node device, that is, to generate a certificate for the node device and perform a AES-GCM key exchange between that device and the Google Cloud application. The Android device uses the MAX66242 as a NFC bridge in order to communicate with the node device’s microcontroller application and ultimately store the certificate into the DS28S60 co-processor, and have the key exchange done between the DS28S60 and the Google Cloud application, using the ECDH protocol. Once this step is achieved, the node device is ready to send its data to the cloud using the negotiated AES-GCM key. Further node authentication by the Cloud is possible using ECDSA since the node now has a valid certificate with a matching key pair. Incidentally, the provisioning process also joins the sensor node to the LoRaWAN network implemented using the ChipStack solution, but this is not the main purpose of the reference design that exhibits a way to secure data without relying on the security of the various underlying communication links.

Features

- Maxim's DS28S60 ChipDNA™ technology protects private and secret keys against invasive attacks.
- Maxim's DS28S60 provides end-to-end security using hardware-based ECDSA authentication, ECDH key exchange and AES-GCM authenticated encryption.
- Complete low-power sensor node board design
- Sample LoRaWAN gateway implementation based on Raspberry Pi
- Sample Google Cloud application showcasing end-to-end security with the sensor board's DS28S60 including ECDH key exchange, and AES-GCM secure communication
- Source code
- Peripheral Module - compatible sensor expansion port
- Raspberry Pi enables portable LoRaWAN Gateway deployment

Hardware Specification

The reference design includes the following major components: DS28S60, MAX32660, MAX66242, DS7505, and SX1262. The DS28S260 is a cryptographic coprocessor that encrypts temperature measurements using an onboard AES-GCM engine, which is accessible through a SPI interface. The DS7505 is a temperature sensor which uses the I2C interface to provide the environment's temperature. The MAX66242 enables the NFC communication between the Android mobile device and the node. The MAX32660 is a Low Power microcontroller which enables the communication between the different modules. Finally, the SX1262 is used to modulate and transmit the encrypted temperature measurement via LoRa protocol.
Designed–Built–Tested
This document describes the hardware shown in Figure 1 as well as its supporting software. It provides a detailed, systematic technical guide to set up and understand the MAXREFDES9001 reference design. The system has been built and tested, details of which follow later in this document.

Figure 1. MAXREFDES9001 hardware.
Quick Start

Required Equipment
- Any PC or notebook computer with an internet browser and a free USB port
- MAX32625PICO board
- Two USB A to USB Micro-B cable
- 10-Pin ARM Cortex Debug cable
- MAXREFDES9001 LoRa End Node Device
- Android Mobile Device
- Raspberry Pi with Dragino PG1301 LoRaWAN Concentrator

Procedure
The reference design, while not available to purchase, was fully assembled and tested by Maxim. Follow the steps below to verify operation:
1) Flash the MAXREFDES9001 board with the node firmware refer to Flashing the Firmware document.
2) Install the MAXREFDES9001 Android Application in the Android device refer to Android Application Deployment document.
3) Setup the Google Cloud and LoRaWAN Gateway refer to Google Cloud and LoRaWAN Gateway Quick-start Guide.

Detailed Description of Hardware
The high-level block diagram of the MAXREFDES9001 hardware is shown in Figure 2. This system is divided into three components: the LoRa end node, the LoRaWAN Gateway, and the Android Device. The LoRa end node includes the following major components: DS28S60 cryptographic coprocessor, DS7505 temperature sensor, MAX32660 low-power microcontroller, MAX66242 NFC Secure Authenticator, and a SX1262 LoRa transceiver. The major components in the LoRaWAN Gateway consist of: A Raspberry Pi and a Dragino PG1301 LoRaWAN Concentrator.

Figure 2. High level block diagram of the MAXREFDES9001.
Detailed Description of Software

The MAXREFDES9001 software is divided into three components: LoRa end node firmware, Cloud Application Software and Android application.

The reference design sequence is as follows:

a) DS28S60 Provisioning of node device
1) The Android Application using the MAX66242 as a communication bridge, triggers the generation of a node device key pair and reads the DS28S60 public key, ROM ID and MANID.
2) The Android Application requests the cloud application to generate a certificate for the node device by submitting the node unique ID and public key.
3) The certificate is returned and stored in the DS28S60 memory.
4) The cloud application provides its public key and is stored in the DS28S60 memory.
5) Using the Diffie-Hellman AES key exchange identical AES-GCM keys are generated in the cloud application and the DS28S60.
6) The cloud application provides LoRa network keys to join the LoRa network and are stored in the node device.

b) DS28S60 Authentication of node device
1) The Android Application requests the DS28S60’s stored certificate and public key.
2) The Android Application requests a challenge from the cloud application providing the certificate and public key.
3) The Android Application requests the DS28S60 to generate an ECDSA signature using the challenge.
4) The Android Application requests the cloud application to verify the ECDSA signature.

c) Data Transmission
1) The node measures temperature using the DS7505.
2) The temperature is encrypted and authenticated using the DS28S60 AES-GCM engine with the exchanged AES key.
3) The secure measurement is sent to the cloud through the LoRaWAN Gateway.
4) The cloud application retrieves the certificate and public keys to authorize the node.
5) The cloud application receives the secure measurement packet.
6) The cloud application retrieves the AES-GCM decryption and verification key associated with the sender node based on the ID located in the measurement packet.

7) The measurement data is decrypted and stored in the Google Cloud database.
8) The database can be browsed from an internet browser to see the measurement.

LoRa End Node Firmware

The LoRa end node firmware consist of four major functions:

a) Provisioning
Consists in requesting the DS28S60 to generate a public key and return its ROMID and MANID and storing the ECC certificate, LoRaWAN keys, AES-GCM peer public key provided by the server in the DS28S60 memory. This is accomplished by writing and reading the MAX66242 memory via I2C the information requested by the application.

b) Authentication
Consists in providing the ECC certificate stored in the DS28S60 memory. A random challenge is received from the server after verifying the ECC certificate authenticity. The DS28S60 performs a Read Page Authentication using the received random challenge and returns the ECDSA signature.

c) Data Encryption
Consists in reading the current temperature using the DS7505 temperature sensor. The DS28S60 generates AES-GCM encryption key by performing a Diffie-Hellman AES key exchange with the AES-GCM keys generated by the server. After performing the AES-GCM encryption using the DS28S60, a ciphertext and authentication tag are generated.

d) Data Transmission
Consists in sending LoRa packets periodically to the LoRaWAN Gateway. The onboard LED flashes indicating the LoRa packets are being transmitted. The LoRa end node joins the LoRaWAN network via Activation by Personalization (ABP) since the LoRaWAN keys are obtained during the provisioning sequence.

There are three types of LoRa packets transmitted sequentially by the node in JSON format.

- **Public Keys packet**: Payload: DS28S60 ROM ID, DS28S60 MAN ID, Public Keys
- **Certificate packet**: Payload: DS28S60 ROM ID, Certificate
- **Encrypted data packet**: Payload: DS28S60 ROM ID, AES-GCM Keys, AES-GCM Ciphertext (Temperature), AES-GCM Authentication Tag
Cloud Application Software
The cloud application software is divided in four components: the ChirpStack network server, ChirpStack application server, Google Cloud App engine, and Google Cloud Datastore.

ChirpStack Network Server
This component routes the received LoRaWAN frames by the LoRaWAN Gateway. It communicates with the ChirpStack Application Server to transmit the received packets by linking them to the corresponding LoRa end node.

ChirpStack Application Server
This component offers a web-interface to add the LoRa end nodes. During the provisioning sequence the LoRa end node is added to the application server, adding the node to the LoRaWAN network as shown in Figure 3. It communicates using the Google Cloud Pub/Sub service to transmit the LoRaWAN packets to the Google Cloud Function.

Google Cloud App Engine
This component performs the provisioning of the LoRa end node. The web application establishes communication with the Android device via WebSocket to generate an ECC certificate based on the DS28S60 public key, and provide the LoRaWAN keys randomly generated. It automatically adds the provisioned LoRa end node to the ChirpStack Application Server.

Google Cloud Function
It processes the data received from ChirpStack Application Server, sorts the LoRa end nodes data based on their unique IDs, retrieves the node’s ECC certificate and public key to authorize it, decrypts the temperature measurement, and stores it in a Google Cloud Datastore.

Figure 3. LoRa end node added to Application Server database after provisioning.
Google Cloud Datastore
This component stores the LoRa end node encryption keys and data. The database is presented as shown in Figure 4.

Android Application
The MAXREFDES9001 Android app serves as a communication interface between the MAXREFDES9001 board and the Google Cloud platform used for this Demo. The purpose of this application is to provide an intuitive interface that can showcase the features of the DS28S60 cryptographic coprocessor by provisioning and authenticating the end node. Through this Android application one can command the DS28S60 to generate the necessary information to register the node to the cloud application.

Main Functions:
The purpose of this demo is to showcase the features of the DS28S60 cryptographic coprocessor for which two main functions were developed.

a) Node Provisioning:
During this process, the DS28S60 is asked to generate a public key from the Android application via the NFC interface that gets stored into the MAX66242’s memory, which is also embedded into the node board. The Android App requests the key and transmits it back to the cloud application through a WebSocket client. The cloud application validates this data and generates a certificate that is returned to the Android App. This data is sent back to the MAX66242 via NFC, which will be then collected by the DS28S60 and will in turn store it.

b) Node Authentication:
This function verifies the authenticity of the node board allowing it to transmit sensor data to the cloud application once the device is authenticated, the cloud application data is stored in the DS28S60. By running this command, the Android application request the DS28S60 to return the certificate generated during the provisioning stage and its public key through the MAX66242 NFC Tag. This data is sent to the cloud application which returns a random challenge that will be used by the DS28S60 to generate a signature which gets sent to the server again to confirm its authenticity.

For more details about the Android Application functionality refer to Android Application Details Document.

Figure 5 shows what the GUI looks like when running on the Android device. See Table 1 for more details on each functionality. See the Design Resources section to download the software and source code.
Figure 5. Main display of the MAXREFDES9001 GUI.
Table 1. GUI Controls

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>FUNCTION NUMBER</th>
<th>DETAILS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Menu</td>
<td>1</td>
<td>Displays the different command options available</td>
</tr>
<tr>
<td>Display Monitor</td>
<td>2</td>
<td>Displays information during the different command processes</td>
</tr>
<tr>
<td>Unauthorized Node Mode</td>
<td>3</td>
<td>Enable or disable sending bogus data simulating the behavior of a counterfeit device</td>
</tr>
<tr>
<td>Command Options</td>
<td>4</td>
<td>List out all the different command options the user has access to</td>
</tr>
<tr>
<td>Node Authentication Command</td>
<td>5</td>
<td>Runs the Node Authentication Sequence</td>
</tr>
<tr>
<td>NFC Provision</td>
<td>6</td>
<td>Runs the NFC Provision Sequence</td>
</tr>
<tr>
<td>Node Provision</td>
<td>7</td>
<td>Runs the Node Provision Sequence</td>
</tr>
<tr>
<td>Registration Info</td>
<td>8</td>
<td>Runs the Registration Info Sequence</td>
</tr>
</tbody>
</table>

Design Resources
Download the complete set of Design Resources including schematics, bill of materials, PCB layout, and test files.
## Revision History

<table>
<thead>
<tr>
<th>REVISION NUMBER</th>
<th>REVISION DATE</th>
<th>DESCRIPTION</th>
<th>PAGES CHANGED</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>11/20</td>
<td>initial release</td>
<td>—</td>
</tr>
</tbody>
</table>

Maxim Integrated cannot assume responsibility for use of any circuitry other than circuitry entirely embodied in a Maxim Integrated product. No circuit patent licenses are implied. Maxim Integrated reserves the right to change the circuitry and specifications without notice at any time. The parametric values (min and max limits) shown in the Electrical Characteristics table are guaranteed. Other parametric values quoted in this data sheet are provided for guidance.

© 2020 Maxim Integrated Products, Inc. All rights reserved. Maxim Integrated and the Maxim Integrated logo are trademarks of Maxim Integrated Products, Inc., in the United States and other jurisdictions throughout the world. All other marks are the property of their respective owners.