AN12257

LoRa 物联网设备安全连接

Rev. 0 — 09/2018

Application Note

目录

1 介绍

很多人都意识到安全连接的重要性，但大家关注的焦点主要是网关或者 IP 设备与互联网的安全连接。网关和终端设备之间的安全连接被忽略了。你可以从一些无线协议（例如 Bluetooth/ZigBee）中找到设备连接的安全性，但是这些安全协议很难复制到其他平台。

本应用说明文档介绍了使用 LoRa 技术如何在网关与终端设备之间建立物联网的安全连接。

所使用的硬件是基于 i.MX RT1050 EVK/LPC845 MAX 和 LoRa 开发板组成的星型网络。安全库软件可以轻松的移植到 NXP 的其他 MCU 平台。

基本固件和设备驱动程序用 C 语言实现，而应用程序层用 C ++语言实现。

2 缩略语

表 1 概述了本文档中使用的缩写。

表 1. 缩略语

缩略语	描述
ACK	Acknowledge
AES	Advanced Encryption Standard
CMAC	Cipher-based Message Authentication
CCM	Counter with Cipher Block Chaining-Message Authentication Code
DES	Data Encryption Algorithm
ECC	Elliptic Curve Cryptography
ECDH	Elliptic-curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
HMAC	Keyed-Hash Message Authentication Code
MAC	Message Authentication Code
SHA1	Secure Hash Algorithm 1
SHA256	Secure Hash Algorithm 256
TLS	Transport Layer Security
ECB	Electronic Cookbook Mode
CBC	Cipher Block Chaining mode
CFB	Cipher Feedback Mode
CTR	Counter Mode
CCM	Counter Mode with CBC-MAC
HMAC	Keyed-Hash Message Authentication Code
MIC	Message Integrity Code
Crt	Certificate. In the document, Crt is just consisted by public key and ECDSA signature value r/s.
CID	Command Identifier
TRNG	True Random Number Generator
NVM	Non-Volatile Memory

3 概述

安全总是以更高的复杂性为代价。因此，只有在真正需要时才应该使用安全机制。何时以及如何使用这些安全机制取决于设备的安全策略。本文介绍了两种增强链路层安全性和构建更高级安全策略的方法。这两种安全方法分别是对称连接和非对称连接。

图 1 所示系统框图。LPC845/i.MXRT1050 通过 SPI 接口控制 LoRa 芯片。这是一个星型网络，i.MXRT1050 作为网关（服务器），LPC845 作为节点（客户端）。网关可以连接 250 个节点。节点之间不能直接通信。实际上，物理层也可以使用其他的通信方式，比如 GFSK。为了安全通信，客户端和服务器端需要建立安全通道。

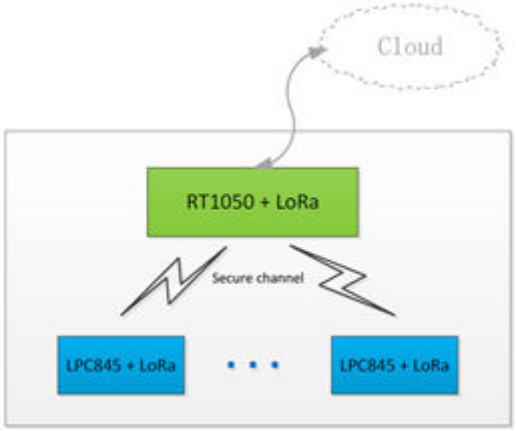

图 1. 系统框图

4 软件架构

软件为三层架构。第一层为用于服务器的 MCUXpresso SDK 软件包和用于客户端的 code bundle 软件包。第二层基于 C++的架构层（由于客户端 LPC845 的资源限制，在客户端程序中删除了此层）。第三层为基于 C++的应用程序。

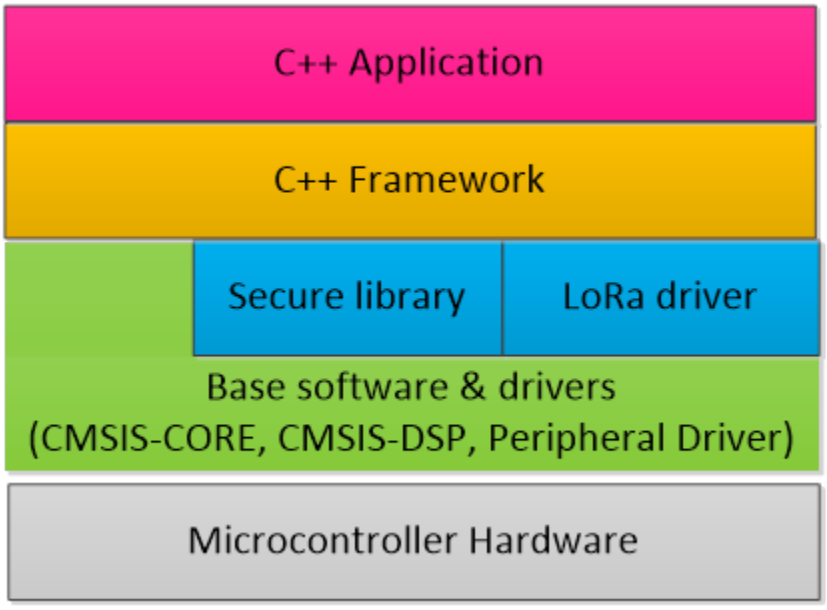

图 2. 软件架构

4.1 安全库

安全库基于 Apache 2.0 许可下的 mbed TLS。mbedTLS 在服务器和客户端向应用程序提供 SSL/TLS 功能，并提供用于构建其他加密协议的加密库。但是我们对 mbedTLS 源代码进行了修改和裁剪，包括必要的物联网安全功能。下面详细介绍下物联网安全库。

- 对称加密算法

  AES, DES

- 操作模式

  ECB, CBC, CFB, CTR, CCM

- 哈希算法

  SHA-1, SHA-224, SHA-256

- MAC 模式

  HMAC-SHA1, HMAC-SHA224, HMAC-SHA256, CMAC

- 椭圆曲线密码（ECC）

  安全库拥有自己的基于 mbed TLS 的大数计算库以用于 ECC 实现，同时支持椭圆曲线 Diffie Hellman（ECDH）和椭圆曲线数字签名算法（ECDSA）。 支持以下标准化曲线/ ECP 组：

  - secp192r1 - 192-bits NIST curve
  - secp224r1 - 224-bits NIST curve
  - secp256r1 - 256-bits NIST curve
  - secp384r1 - 384-bits NIST curve
  - secp521r1 - 521-bits NIST curve
  - secp192k1 - 192-bits Koblitz curve
  - secp224k1 - 224-bits Koblitz curve
  - secp256k1 - 256-bits Koblitz curve
  - bp256r1 - 256-bits Brainpool curve
  - bp384r1 - 384-bits Brainpool curve
  - bp512r1 - 512-bits Brainpool curve
  - m255 - 255-bits Curve25519

- 随机数生成

  我们提供 NIST 标准化的 HMAC_DRBG 伪随机数生成器。

  可以找到基于 MDK 开发工具的 IoT 安全库（RT1050_Security_lib.lib 和 LPC84x_Security_lib.lib 工具）。

4.2 C ++框架

如 图 3 所示，建立了一个用于 C ++项目的框架。 四个主要基础类：

- CSystem : 包括一个任务调度器。 任务的类型包括高优先级任务，低优先级任务和切片任务。 如果任何类要创建一个任务，则需要继承 COneTask 类并注册。
- CHardware: 包括与硬件相关的类，例如 Radio 类。
- CCommunication: 包括与通讯相关的类，例如 secureMAC 类。
- CBusiness: 包括相关或虚拟对象。例如，当接收到客户端加入命令时，将创建一个客户端对象，而在发送/接收断开连接命令时，将删除一个相对应的客户端对象。 KeyManagement 类包含在 CBusiness 类中。

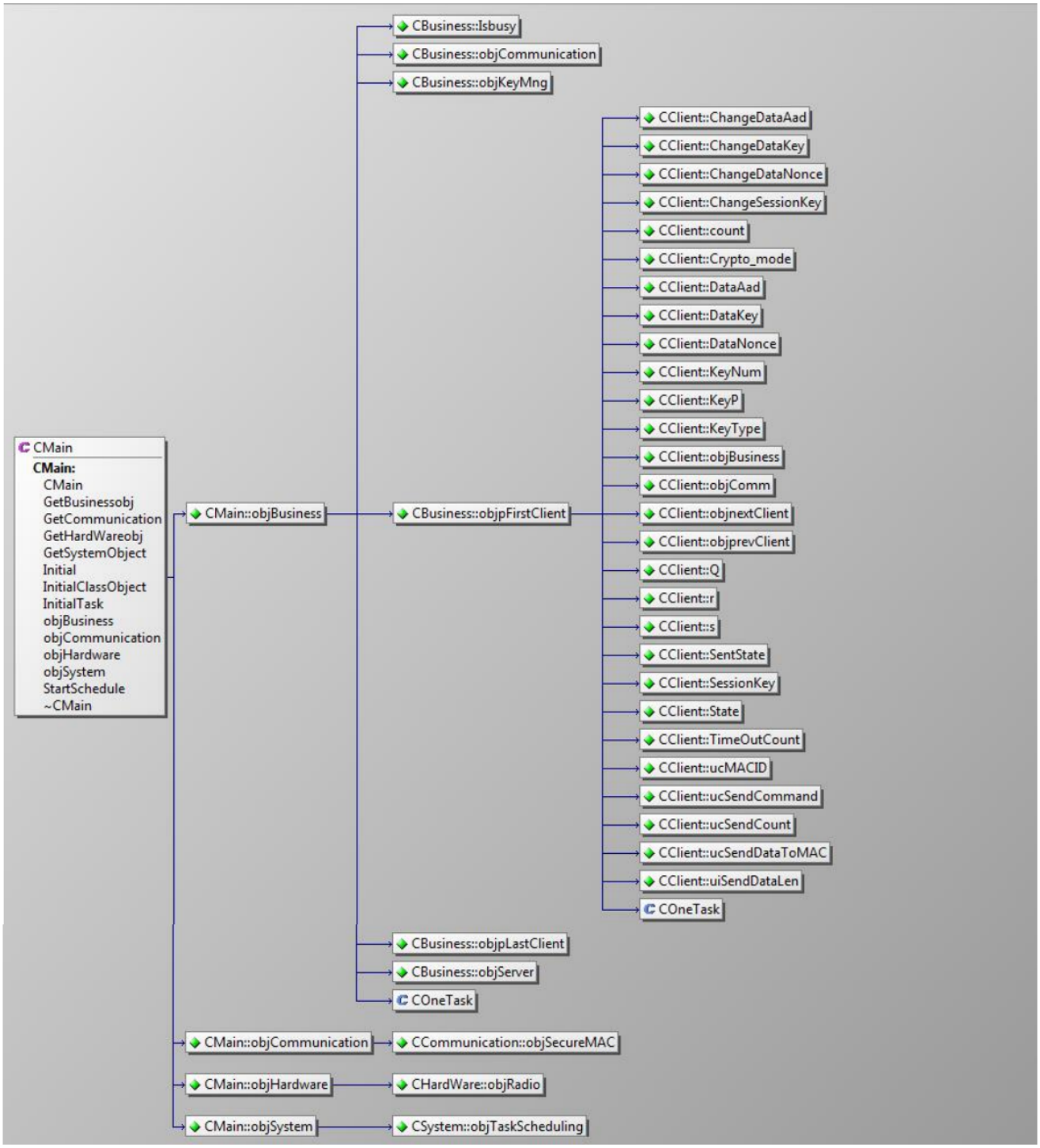

图 3. C ++框架

5 安全链接层消息格式

为安全连接添加了安全链路层，所有的上行和下行消息都以 Link Header 开始，紧接着是两个 Link ID(From Link ID 和 To Link ID), 然后是链路层有效负载（PayLoad）,最后一 MIC 验证码结束。

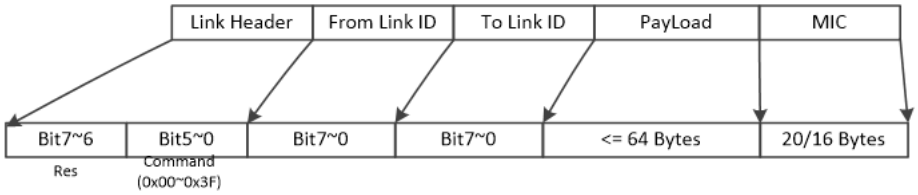

图 4. 安全链路层消息格式

5.1 Link Header

Link Header 的 Bit7-6 位是保留位， Bit5-0 代表特定的消息命令。 表 2 列出了详细的命令。

表 2. 链接命令

CID	命令	发送者		描述
		客户端	服务器	
0x01	Join Request	√		用于客户端发起加入网络请求
0x02	Join Accept		√	回答 Join Request 命令。 服务器允许该客户端加入网络。
0x03	Join Type	√		Join Type 包含两种：对称和非对称。 从有效载荷消息中判断使用的类型。
0x04	Type Accept		√	回答 Join Type 命令。
0x10	Key Type	√		如果 Join Type 是对称的，则 CID（0x10-0x14）可用。客户端需要将自己的密钥发送到服务器。
0x11	Key number		√	回答 Key Type 命令，服务器确认密钥类型正确。 然后服务器发送 Key number 命令并告诉客户端将使用哪个密钥。
0x12	Key confirm	√		回答 Key number 命令。告诉服务器密钥确定。
0x13	Symmetric session key		√	服务器生成会话密钥，然后将其发送给客户端。

Table continues on the next page...

表 2. 链接命令 (continued)

CID	命令	发送者		描述
		客户端	服务器	
0x14	Symmetric session key confirmed	√		回答 Symmetric session key 命令。
0x20	Request server Crt	√		如果 Join Type 为非对称，则 CIDs（0x20-0x28）可用。 客户端请求服务器的 Crt。 需要几次传输才能获得 Crt。
0x21	Response server Crt		√	回答 Response server Crt 命令。 服务器将自己的 Crt 发送给客户端。 需要多次发送 Crt。
0x22	Server Crt confirmed	√		收到服务器 Crt 后，客户端将验证 Crt。 如果 Crt 正确，则客户端将发送此命令。
0x23	Request client Crt		√	服务器请求客户端的 Crt。 需要多次才能获得 Crt。
0x24	Response client Crt	√		回答 Response client Crt 命令。客户端将自己的 Crt 发送到服务器。 需要多次发送 Crt。
0x25	Client Crt confirmed		√	收到客户端 Crt 后，服务器将验证 Crt。 如果 Crt 是正确的，服务器将发送此命令。
0x26	Shared Key confirmed	√		确认两个 Crt 后，客户端和服务器将生成相同的共享密钥。 客户端发送此命令。
0x27	Asymmetric session key		√	服务器生成会话密钥，然后将其发送给客户端。
0x28	Asymmetric session key confirmed	√		客户端确认会话密钥已成功接收。

Table continues on the next page...

表 2. 链接命令 (continued)

CID	命令	发送者		描述
		客户端	服务器	
0x30	Data Key		√	服务器生成数据密钥，然后将其发送给客户端。
0x31	Data Key confirmed	√		客户端确认数据密钥已成功接收。
0x32	Security connected		√	服务器通知安全通道已建立。
0x35	Change Session Key	√	√	客户端或服务器可以发送命令以请求更改会话密钥。
0x36	Change Session Key confirmed	√	√	回答 Change Session Key 命令。 接收方生成新的会话密钥，并将其发送到请求的一侧。
0x37	Change Data Key	√	√	客户端或服务器可以发送命令以请求更改数据密钥。
0x38	Change Data Key confirmed	√	√	回答 Change Data Key 命令。接收方生成新的数据密钥，并将其发送到请求的一侧。
0x3A	Data	√	√	建立安全通道后，将有用的数据发送到另一侧。
0x3B	ACK	√	√	回答 Data 命令。 接收方发送命令告诉发送方消息已接收。
0x3F	Disconnect	√	√	发送命令以断开安全通道。

5.2 Link ID

安全链接层中有两个 Link ID。

- 一个 Link ID 为 From Link ID，是指消息的来源；
- 另一个 Link ID 为“To Link ID”，是消息的去处。

当收到一个包时，首先需要根据 To LinkID 来判断该包是否是给“我”的，以及是否需要根据 From Link ID 来处理该包。

5.3 Payload

如果数据帧承载有效负载（PayLoad），则可能在计算消息完整性代码之前对 PayLoad 进行加密，可以从 PayLoad 中获取有用信息。 有关如何加密/解密有效负载的信息，请参阅应用层数据加密。

5.4 MIC

消息完整性代码（MIC）是在消息中的所有字段上计算得到的。

```
msg = Link Header | From Link ID | To Link ID | PayLoad
```

有两种执行消息完整性代码（MIC）的方法，包括 HMAC-SHA1 和 CMAC。

- `HMAC-SHA1(K, msg) = H((K ⊕ opad )|| H((K ⊕ ipad) || msg))`

  其中：

  — K 是密钥。

  — H 是 Hash 函数。

  — ⊕是异或运算

  — iPad 是内部填充。

  — opad 是外部填充。

  MIC =HMAC-SHA1[0…19], 其中 MIC 应该是 20 个字节。

- `CMAC = aes128-cmac(K, msg)`

  MIC =CMAC[0…15], 这里 MIC 应该为 16 个字节。

6 密钥管理

密钥管理（Keymanagement）涉及加密密钥的生成、分发、存储和处理。安全连接中涉及许多密钥。由于在建立安全通道期间密钥用于加密/解密消息，因此密钥的存储非常重要。表 3 列出了所有使用的密钥。

表 3. 密钥简介

密钥名 [1]	生成方法	链路层命令(CID)	使用域(LINK LAYER MIC/DATA ENCTYPTION)	描述
HKey[2]	Stored[3]	0x01-0x26	LINK LAYER	客户端和服务器应具有相同的 Hkey。存储在非易失性存储器中。
ECCKey[4]	Stored	—	—	如果连接模式为“非对称”，则服务器和客户端应存储自己的 ECC Key。
SymmKey[5]	Stored	—	—	如果连接模式为“对称”，则客户端存储 5 个密钥，而服务器也存储相同的客户端密钥。
Keyx	Stored	0x13-0x14>, 0x30-0x32	DATA ENCRYPTION	该密钥是 SymmKey 之一。 如果连接模式为

Table continues on the next page...

表 3. 密钥简介 (continued)

密钥名 [1]	生成方法	链路层命令(CID)	使用域(LINK LAYER MIC/DATA ENCTYPTION)	描述
				“对称”，则服务器将随机选择一个 SymmKey 作为临时数据密钥。
Keyp	Server/Client	0x27-0x28, 0x30-0x32	LINK LAYER MIC	验证 Crt 后，服务器和客户端将获得彼此的公钥。 服务器和客户端通过 ECDH 算法生成相同的共享密钥 Keyp。
Session key	Server	0x30-0x3F	LINK LAYER MIC	在建立安全通道期间，服务器使用 TRNG 生成会话密钥。在收到 Change Session Key 时，服务器和客户端也会产生该密钥。改会话密钥”命令，则服务器或客户端会生成它。
Data key	Server/Client	0x35-0x3F	DATA ENCRYPTION	在建立安全通道期间，服务器使用 TRNG 生成会话密钥。在收到 Change Session Key 时，服务器和客户端也会产生该密钥。

1. 密钥管理与整个系统的安全性有关，因此我们只列出使用的密钥。如果您想了解更多信息，可以在 NXP Secure MCU 功能中进行查看，例如 PUF/OTP/DICE…
2. 例如，在 security_sw_RT 和 security_sw_LPC84x 工程中的 ConfigutationInfo.c 文件中，ConstHKey 数组用于此目的
3. Stored 是指在系统工作之前，密钥已存储在非易失性存储器中。所有存储的密钥都需要事先注入 NVM，最好将其存储在 OTP 存储器中。

   安全通道断开连接后，所有未存储的密钥都需要销毁。一定时间后需要更改会话密钥和数据密钥。
4. 例如，在 security_sw_RT 和 security_sw_LPC84x 工程中的的 ConfigurationInfo.c 文件中，Serverd/Clientd 数组是 ECC 私钥，而 Q_XYZ 的数组是 ECC 公钥。
5. 例如，在 security_sw_RT 工程的 ConfigurationInfo.c 文件中，ConstsymmKey 数组由 5 个客户端的 SymmKey 组成，每个客户端 SymmKey 由 5 个密钥组成。

7 应用层数据加密

对安全链接层的有效负载（PayLoad）的加密，称为应用程序层数据加密。根据不同的 CID，有两种不同的加密/解密方法：AES128-CBC，AES128-CCM。

7.1 AES128-CBC 加密/解密

有效负载是由 AES128-CBC 加解密。此加密方式仅用于在建立安全通道期间交换会话密钥和数据密钥期间，CID 为 0x13-0x14、0x27-0x28 和 0x30-0x32 的包是由 AES128-CBC 加密的。

7.2 AES128-CCM 加密/解密

有效负载是由 AES128-CCM 加密和认证。 CCM 通过结合使用 Counter（CTR）模式技术和 CBC-MAC 算法来确保数据的机密性和真实性。建立安全通道后使用此加密/解密方式，CID 为 0x35-0x3F 的包使用此方式。对纯文本进行加密后，将生成具有相同长度的加密数据和 8 个字节标记。

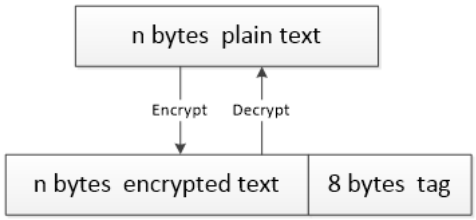

图 5. 消息加密和认证

8 建立安全通道的过程

在客户端与服务器进行正式的通信之前，应建立安全通道。建立安全通道的方法有两种：对称连接和非对称连接。两种方法都使用相同的启动连接和安全的数据交换过程。

8.1 启动连接

安全连接由客户端发起。服务器通过 Hkey 验证软件包的完整性，通过判断 `From Link ID` 判断是否允许客户端加入网络。客户端收到 `Join Accept` 命令后，将发送 `Join Type` 通知服务器建立安全通道的方法（通过有效载荷的第一个字节判断：0x00（对称连接），0x01（非对称连接））。

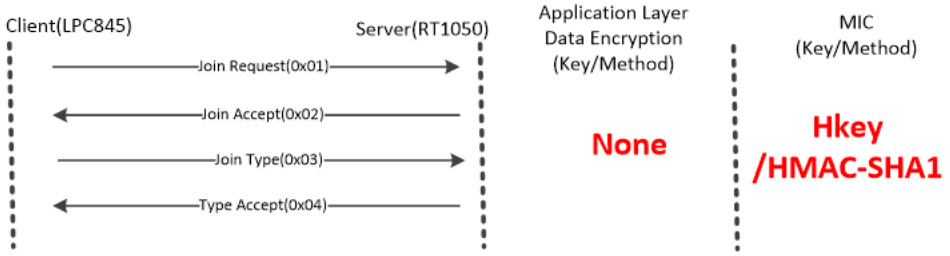

图 6. 启动连接

8.2 对称连接

客户端收到 `Type Accept` 命令后，客户端发送 `Key Type` 命令告诉服务器可以使用五个 SymmKey。 服务器使用 TRNG 生成随机数以选择一个 SymmKey（Keyx）并告知客户端密钥号。 然后，服务器通过 TRNG 生成会话密钥，并以 CBC 模式将其通过 Keyx 加密发送。

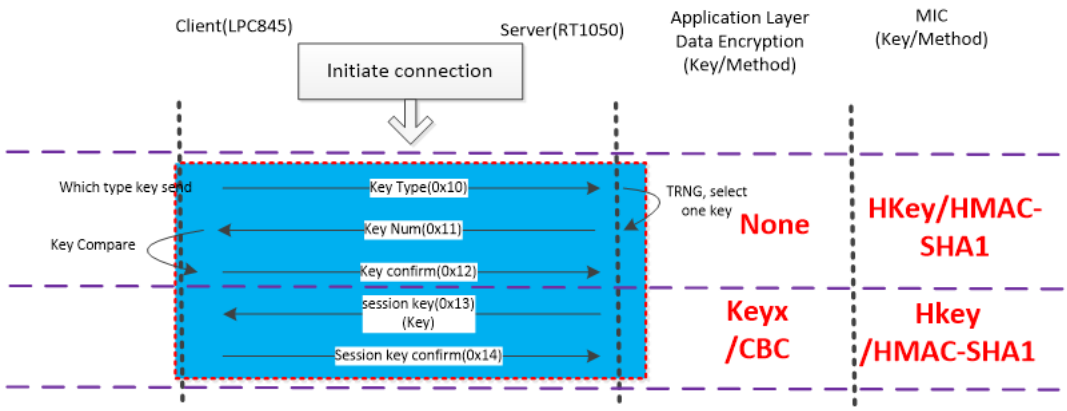

图 7. 对称连接

8.3 非对称连接

客户端收到 `Type Accept` 命令后，客户端和服务器将获得对方的 Crt 并由 ECDSA 进行验签。然后它们可以通过 ECDH 获得相同的共享密钥（keyp），并使用它来加密/解密应用层数据。会话密钥由服务器的 TRNG 生成。

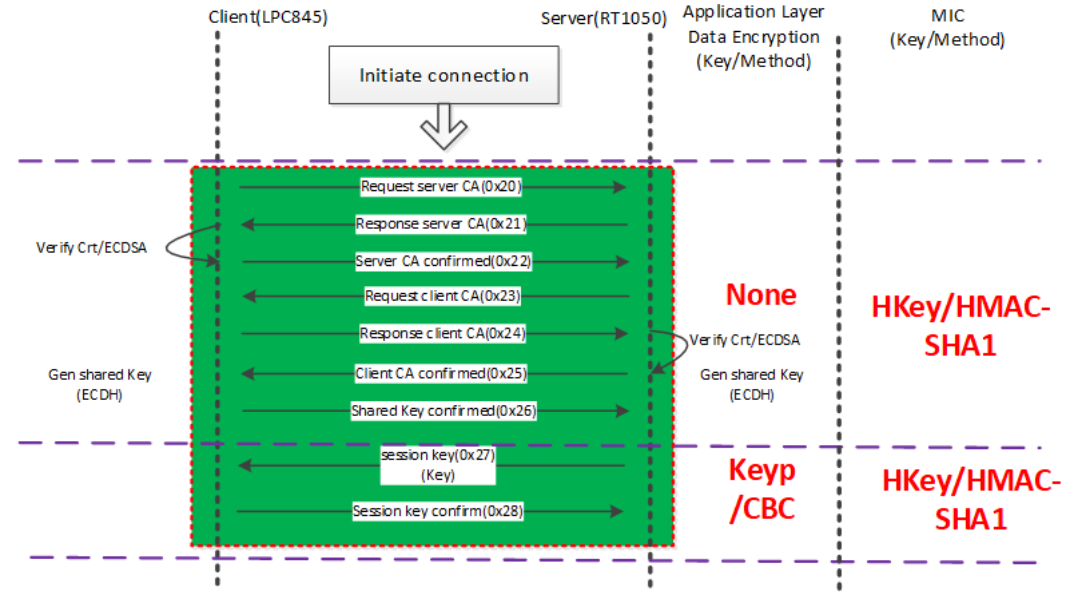

图 8. 非对称连接

8.4 安全的数据交换

会话密钥生成后，MIC 由 CMAC 模式下的会话密钥计算。服务器通过 TRNG 生成数据密钥（包括密钥，随机数，AAD），并在 CBC 模式下将其通过 Keyp 或 Keyx 加密发送。当客户端收到 CID 为 0x32 的命令时，将建立安全通道。服务器和客户端可以在安全模式下使用 CID 为 0x3A 和 0x3B 的命令进行通信。 CID 为 0x3F 的命令表示发送方请求断开安全连接。

注意

一段时间后，应更新会话密钥和数据密钥。

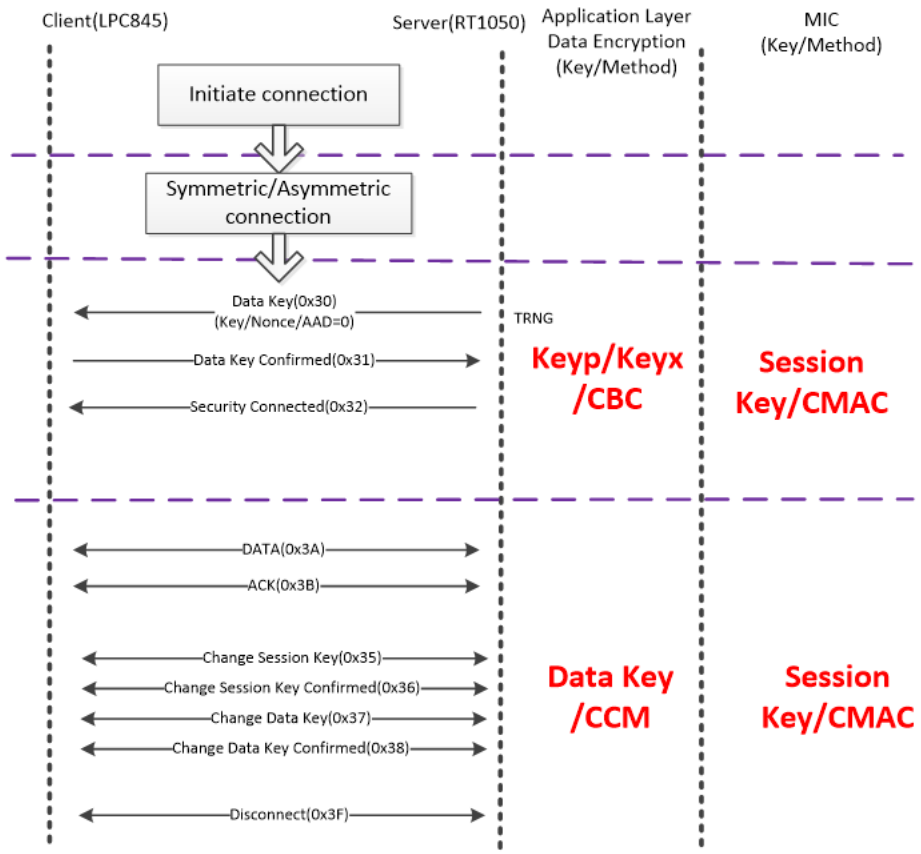

图 9. 安全数据交换

9 硬件平台

本章节介绍了演示应用程序的硬件平台。

9.1 LoRa 板

LoRa 板卡包含 LoRa 模块和具有 Arduino 接口的 LoRa 底板。

- LoRa 模块的主要功能：
  - 无线电频率：400-500 MHz
  - 高达+20 dBm 的恒定 RF 输出
  - 高灵敏度：低至-135 dBm
  - 前导检测
  - 最大发送包 256 字节，带有 CRC 校验
- LoRa 底板的主要功能：
  - Arduino 接口
  - 用户 LED 显示 LoRa 模块的状态
  - 温度传感器：PCT2075
  - SMA 接口

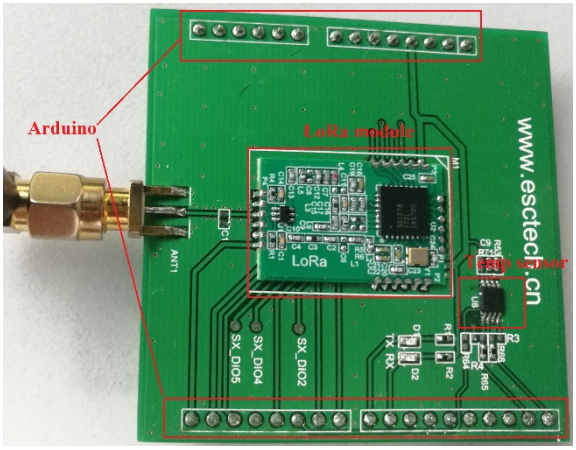

图 10. LoRa 板

9.2 服务器硬件

i.MX RT1050 EVK 开发板是 LoRa 板卡的底板。

9.2.1 i.MXRT1050 EVK 开发板介绍

i.MXRT1050 EVK 的主要功能：

- 内存：256 Mbit SDRAM，64 Mbit Quad SPI Flash，512 Mbit Hyper Flash，TF 卡插槽。
- 通讯接口：USB 2.0 OTG 连接器，USB 2.0 主机连接器，10/100 M 以太网连接器，CAN 总线连接器。
- 多媒体接口：CMOS 传感器连接器，LCD 连接器。
- 音频接口：3.5 毫米立体声耳机插孔，板载麦克风，SPDIF 连接器（默认情况下未安装）。
- 硬件和软件平台。
- 调试接口：带有 DAP-Link，JTAG 20 针连接器的板载调试适配器。
- Arduino 接口。
- 用户按钮和 LED 灯。

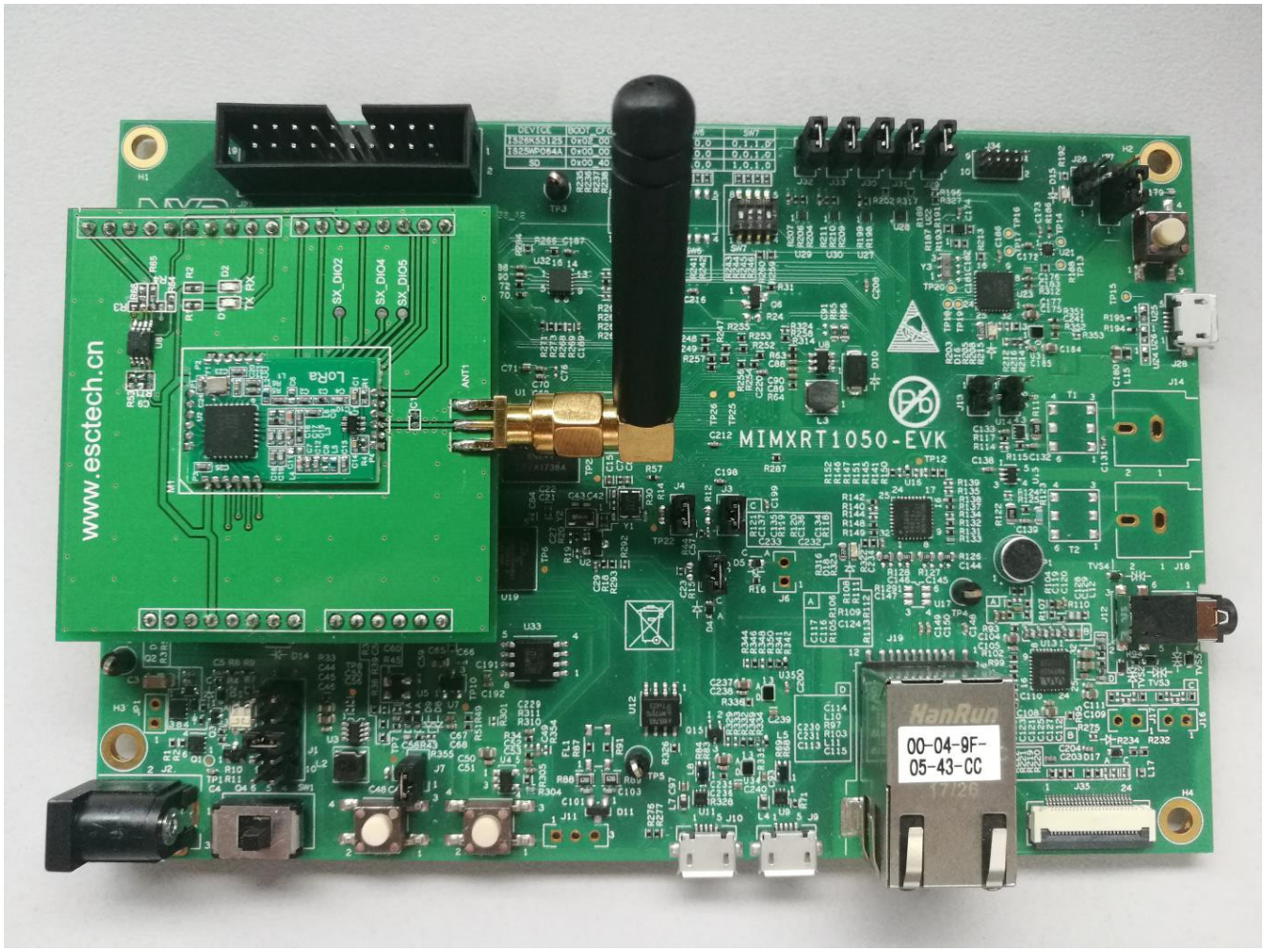

图 11. 服务器硬件

9.2.2 i.MXRT1050 EVK 开发板设置

为了启用 spi 和某些 GPIO 功能，需要更改 EVK 板设置。

- 拆下电阻：R341
- 焊接电阻：R278，R279，R280，R281，R288，R289，R276，R277

更改这些设置后，i.MXRT1050 EVK 板和 LoRa 板即可工作。

9.3 客户端硬件

LPC845 MAX 开发板是 LoRa 板的控制板。LPC845MAX 板的主要功能是：

- 基于 LPC11U35 MCU 的板载 CMSIS-DAP 调试器带有 VCOM 端口。
- 允许使用外部的调试器调试目标 MCU。
- 红绿蓝三色用户 LED 灯。
- ISP 接口和用户/唤醒模式按钮。
- MCU 复位按键。
- LPCXpresso 扩展连接器。

- 通过扬声器驱动的 DAC 输出。
- 与 Arduino UNO 平台兼容的 Arduino™连接器。

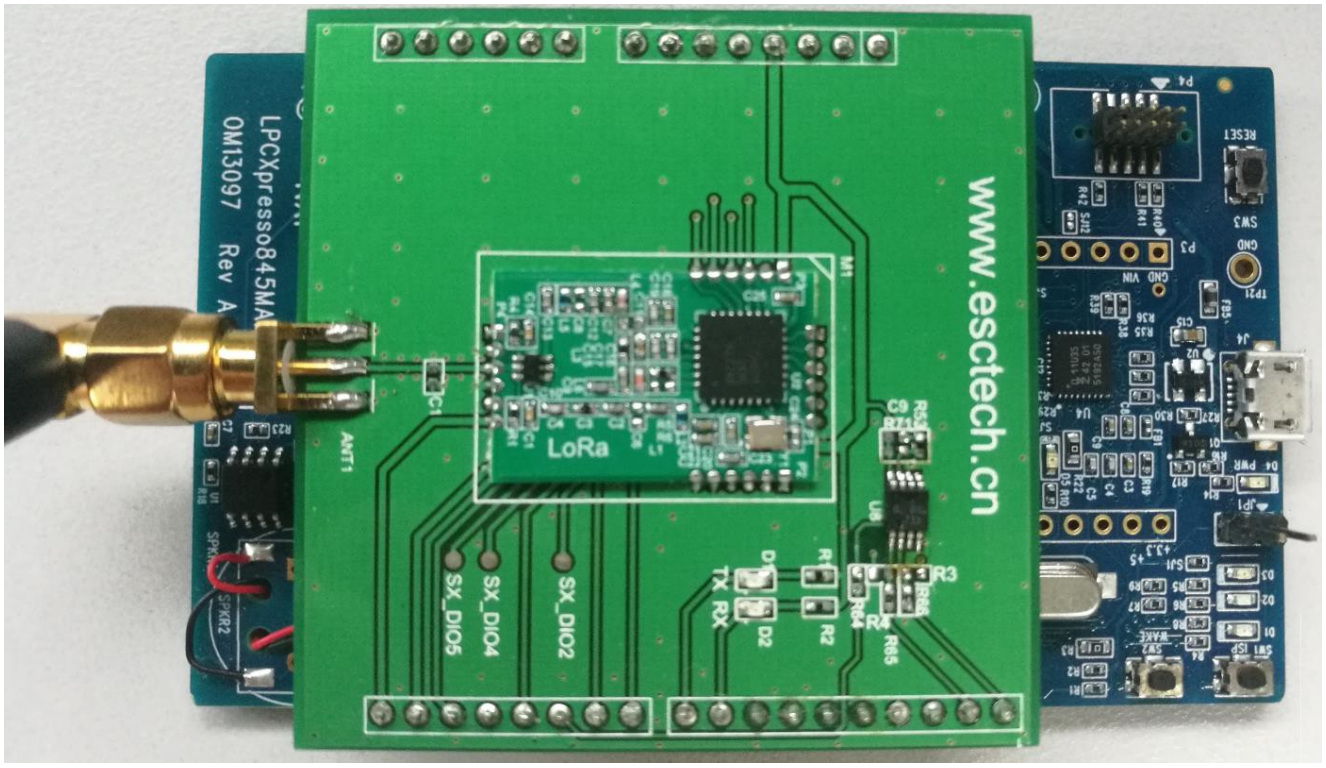

图 12. 客户端硬件

无需修改 LPC845 MAX 板即可适用于 LoRa 板卡。

10 软件平台

软件平台有两种：一种是基于 i.MXRT1050 SDK 的服务器软件平台，另一种是基于 LPC845 code bundle 的客户端软件平台。应用程序代码用 C++语言实现。开发工具链为 Keil MDK 5.24。

10.1 服务器软件

服务器代码实现了非对称客户端和对称客户端入网的功能。相关设备信息可在 ConfigurationInfo.c 文件中找到。

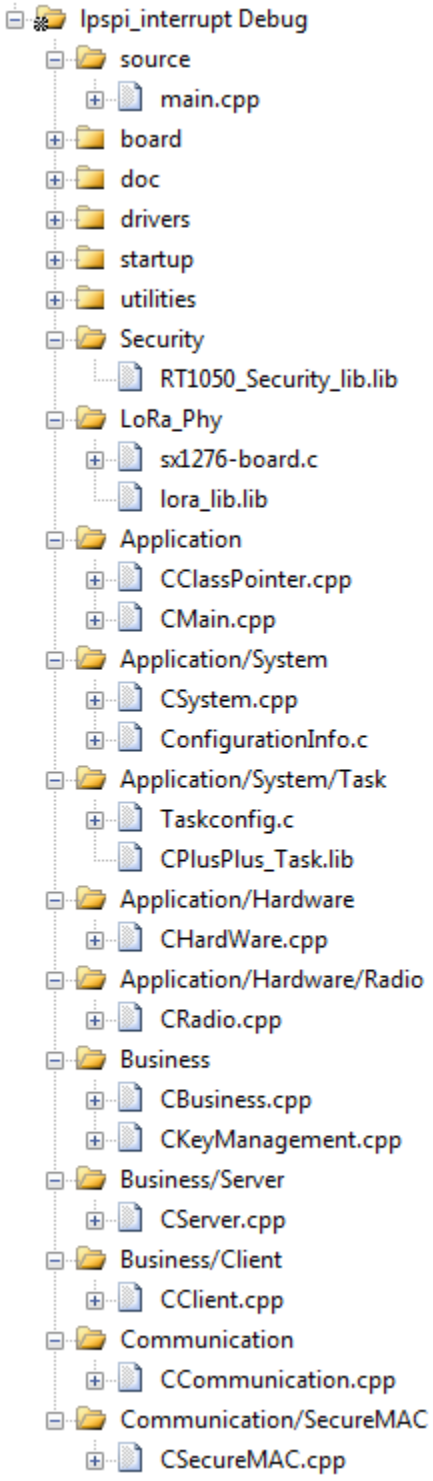

图 13. 服务器软件

10.2 客户端软件

由于 LPC845 的资源有限，因此删除了 C ++框架，并且没有任务调度器。

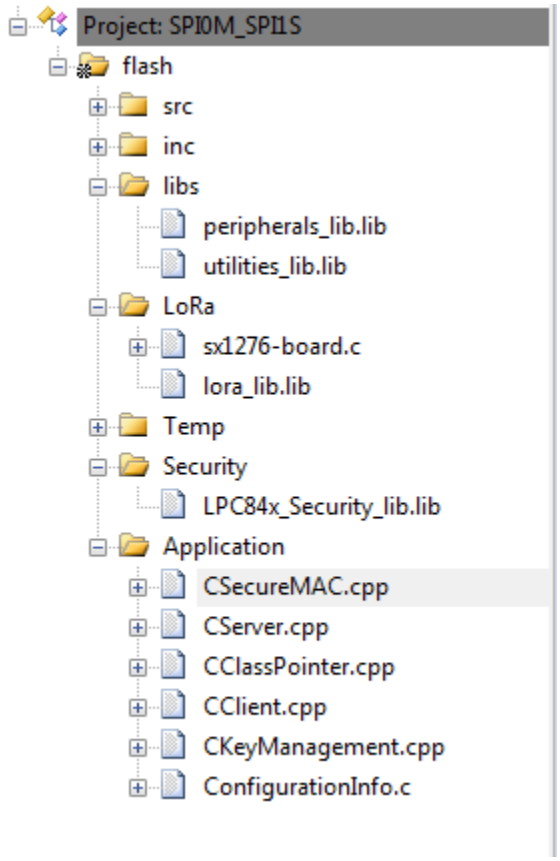

图 14. 服务器软件

11 动手实验

在此动手实验中，具有非对称连接功能的客户端 1 和具有对称连接功能的客户端 2 将被加入到星形网络中。参见 图 15

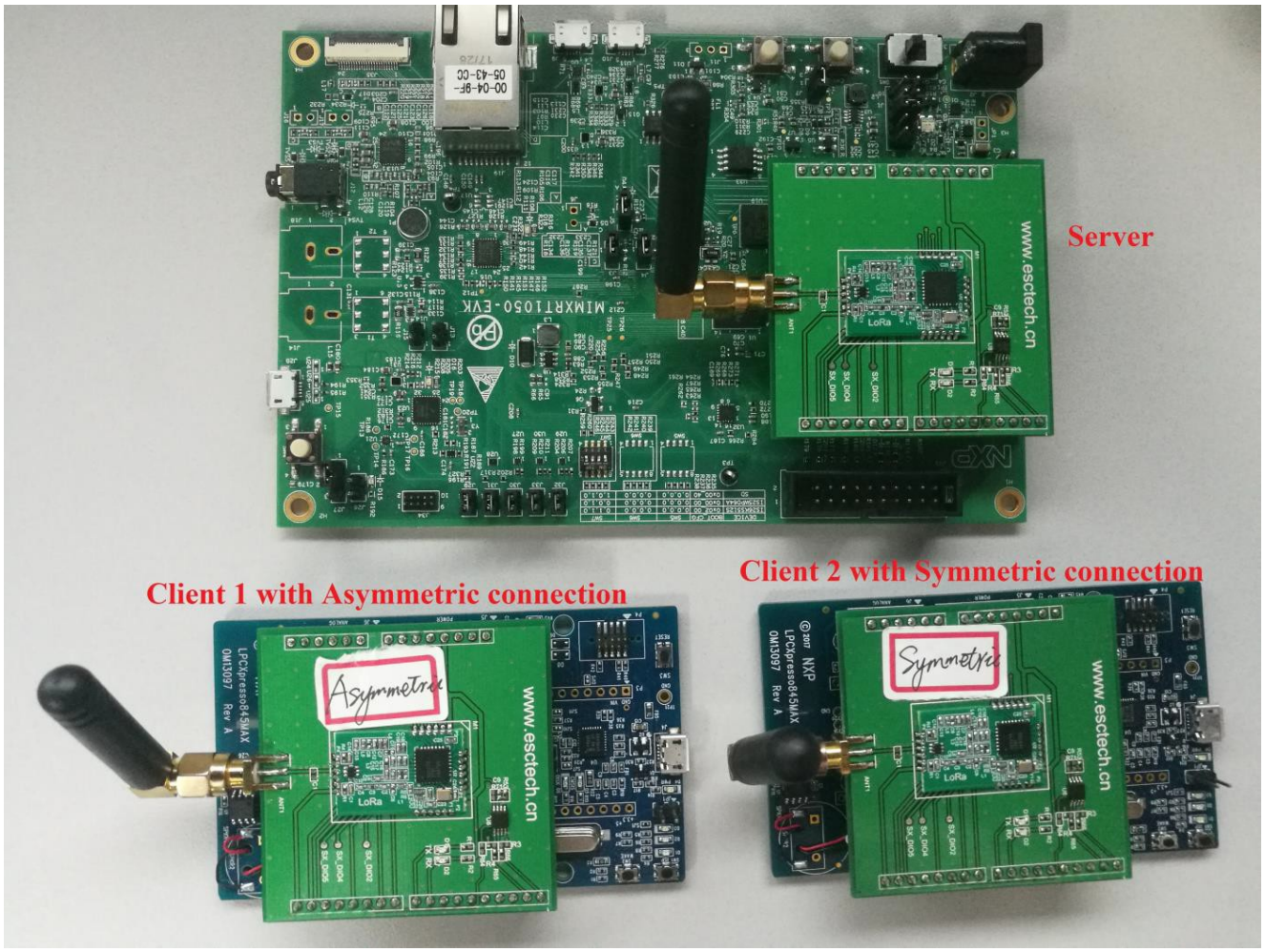

图 15. 演示板

要使这三个板正常工作，请执行以下步骤：

1. 用 micro USB 电缆连接 PC 主机和三个板卡的 OpenSDA USB 端口。
2. 打开串口终端，在 PC 上为 OpenSDA 串口设置如下：
   - 115200 波特率
   - 8 个数据位
   - 1 位停止位
   - 无流量控制
3. 编译相关工程，然后将程序下载到目标板上。

   注意

   如果是对称连接，请记住定义 `DefSYMMETRIC`。

4. 按下三个板卡上的的复位按键。

   代码成功运行后，可以从终端上看到类似的信息，如 图 16 所示是非对称连接。

图 16. 非对称连接

图 17 显示对称连接。

图 17. 对称连接

12 用例

本篇应用笔记涉及安全库以及如何建立安全通道，因此在不同的情况下可以复用。 我们列出了一些用例。

12.1 点对点安全连接

两个无线或有线设备在建立安全通道模式下进行通信。

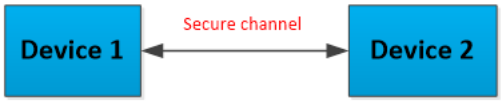

图 18. 点对点安全连接

12.2 安全的星型/mesh 网络

局域网中，有时需要创建一个安全网络来防止中间人攻击。 例如。 LoRa/GFSK 网络没有标准的安全协议，因此复制此应用笔记的做法是一个适当的选择。

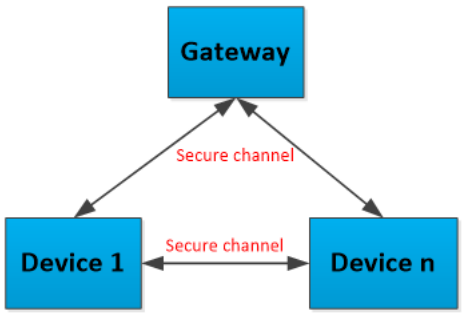

图 19. 安全的星/网网络

12.3 RS484/CAN 的安全连接

在工业领域，RS485/CAN 是一种底层协议，不支持任何安全功能。 在大多数实现中，期望应用程序部署其自己的安全性机制，为此我们给出了一个示例。

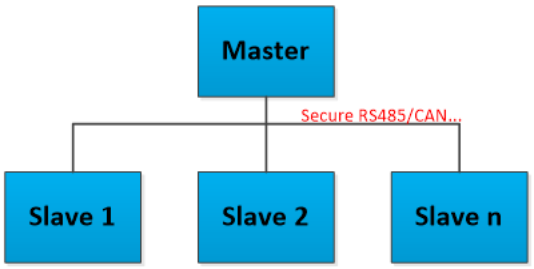

图 20. RS485/CAN 的安全连接

13 结论

本应用笔记介绍了如何在设备之间建立安全连接。 为 NXP MCU 客户提供了 IoT 安全库。可以为相关产品使用类似的安全连接。

如果您对该参考设计的硬件和软件感兴趣，请发送电子邮件到市场团队索取相关资料。

14 参考

- Bluetooth Security
- LoRaWAN Specification
- MBED
- ARM Cortex-M7 Processor Technical Reference Manual

- *i.MX RT1050 Processor Reference Manual* (document IMXRT1050RM)
- LoRaSX1276/77/78/79 data sheet

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 09/2018

Document identifier: AN12257

arm