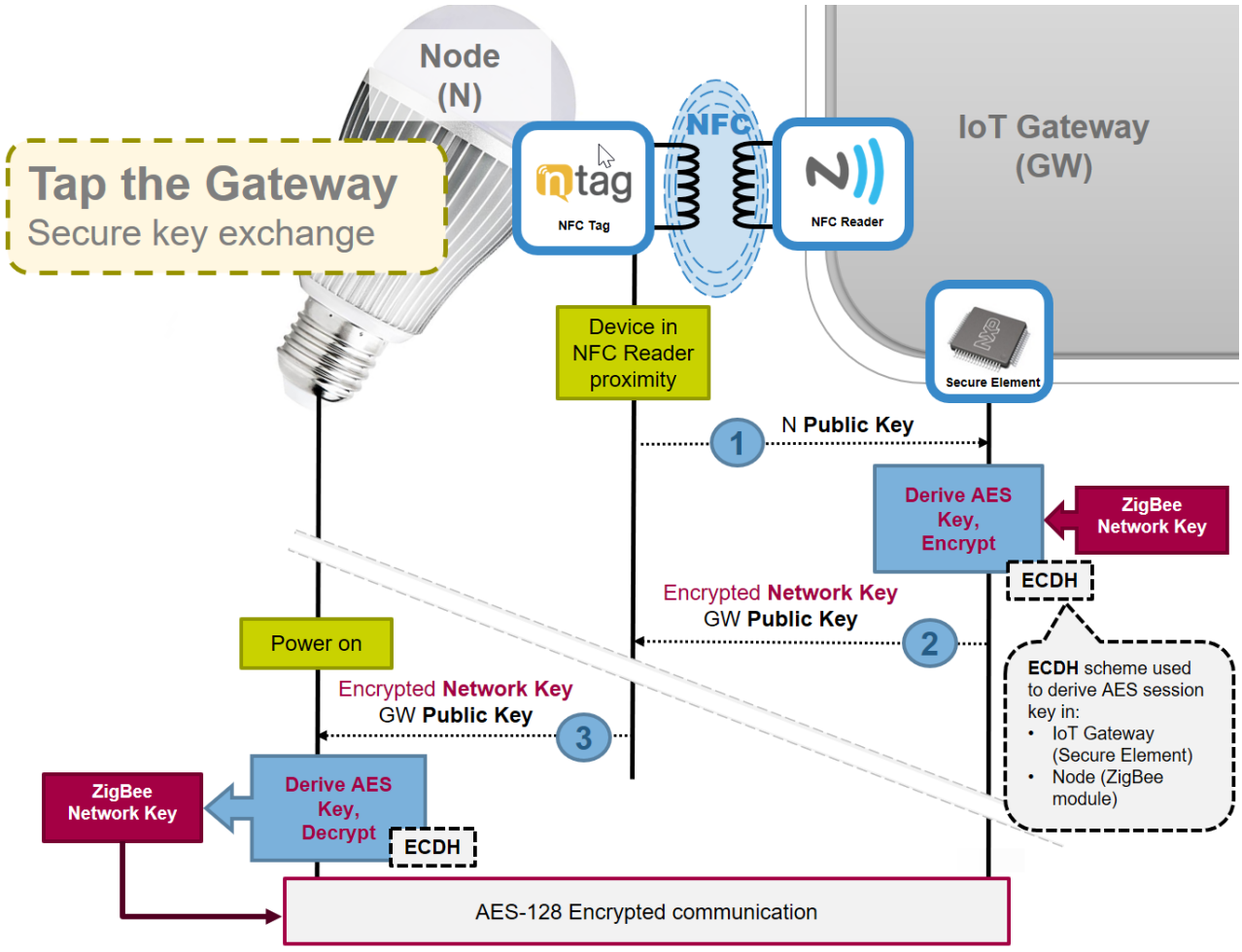
NXP ZigBee3.0 使用 NTAG I²C 实现安全入网

(shaozhong.liang@gmail.com)

在 NXP ZigBee3.0 应用中，我们可以通过使用带外技术来传输网络凭据，让设备加入网络和进行密钥交换的标准方法得以增强。使用连接到 ZigBee 设备的 NFC NTAG I²C 标签，可以实现安全的信息传输，而不会被恶意无线网络侦听器检测到。

NXP NT3H2x11 NTAG I²C 解决方案将无源 NFC 接口与 I2C 接口相结合。 NT3H2x11 设计用于家庭自动化、消费类应用，是添加一触即通连接的快速而经济的解决方案。NTAG I²C 标签充当双端口存储器，一个端口连接到 NFC 接口，另一个端口连接到 I2C 端口。数据可从 NFC 读写器通过 13.56MHz 无线方式传输到 NTAG 存储器，然后 ZigBee 网络的无线微控制器使用 I2C 接口读取该信息。在数据传输过程中无需为 ZigBee 网络的无线微控制器供电。例如下图 1 所示的 NXP NTAG 器件，可以存储 1KB 或 2KB 数据。



NTAG I²C for ZigBee 解决方案所需的硬件。树莓派 Raspberry + PN7120 NFC Reader 组成 ZigBee Gateway。NXP 最新的 JN-518x 系列 ZigBee SoC 已经集成片内 NTAG 功能，只需外接 NFC 天线即可。

最新的 NXP ZigBee3.0 SDK 已经提供了 NFC 驱动代码。下面是 NFC 功能主要接口 API 函数。

```
PUBLIC void NTAG_vInitialise(
       uint8    u8Address,     /*!< Reader I2C address (0xFF for automatic detection) */
       uint8    u8I2cScl,      /*!< 6x: 16   = DIO16
                                *       others = DIO14
                                *   7x: 4      = DIO4
                                *       others = DIO3
                                *   8x: 6      = I2C1 DIO6
                                *       12     = I2C1 DIO12
                                *       15     = I2C0 DIO15
                                *       0xff   = I2C2 (internal NTAG)
                                *       others = I2C0 DIO10 */
       uint32  u32FrequencyHz, /*!< Frequency in Hz */
       uint8    u8InputFd      /*!< Input DIO for field detect
                                *   0xff to ignore FD line */
       );

PUBLIC  bool_t NTAG_bRead(
        uint32    u32ReadAddress, /*!< Byte address of data to read */
        uint32    u32ReadLength,  /*!< Number of bytes to read */
        uint8    *pu8ReadData     /*!< Buffer to read data into */
        );

PUBLIC  bool_t NTAG_bWrite(
        uint32    u32WriteAddress,  /*!< Byte address of write */
        uint32    u32WriteLength,   /*!< Number of bytes to write */
        uint8    *pu8WriteData      /*!< Buffer to write data from */
        );

PUBLIC void NTAG_vRegCbEvent(
       tprNtagCbEvent prRegCbEvent /*!< Pointer to event callback function */
       );

PUBLIC void NTAG_vTick(
       uint32  u32TickMs /*!< Number of ms since previous call */
       );

PUBLIC teNtagNwkStatus NTAG_NWK_eRead(
    uint32          *pu32ReadAddress,         /*!< address of NTAG NWK NDEF */
    tsNfcNwkPayload   *psNfcNwkPayloadStart   /*!< NTAG NWK NDEF payload data */
    );

PUBLIC teNtagNwkStatus NTAG_NWK_eWrite(
    uint32          *pu32WriteAddress,      /*!< Pointer to byte address to write data */
    tsNfcNwkPayload   *psNfcNwkPayloadStart /*!< Pointer to payload to write */
    );

PUBLIC teNtagNwkStatus NTAG_NWK_eTick(
       uint32                u32TickMs /*!< Time in ms since previous call */
    );
```

由于通过 I2C 方式进行 NTAG 读、写速度比较慢，必须通过异步方式进行读写。因此需要启动一个 5ms 的定时器，驱动 NTAG 的读、写状态机。通过 NTAG_vRegCbEvent 注册的事件回调函数获得 NTAG_bRead 读操作、NTAG_bWrite 写操作的执行结果。

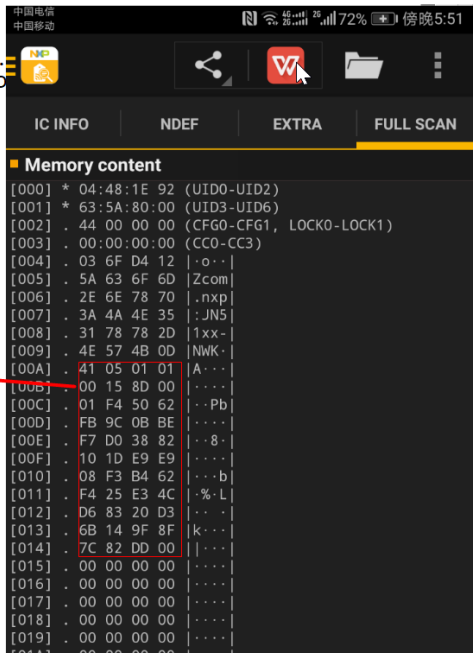
```
PUBLIC  void        APP_cbNtagEvent(    /* Called when an event takes place */
        teNtagEvent eNtagEvent,         /* Event raised */
        uint32      u32Address,
        uint32      u32Length,
        uint8       *pu8Data)           /* Event data (NULL if no data) */
{
    /* Which event ? */
    switch (eNtagEvent)
    {
        /* Absent ? */
        case E_NTAG_EVENT_ABSENT:
        {
            /* Add customer’s code here */
        }
        break;

        /* Present ? */
        case E_NTAG_EVENT_PRESENT:
        {
            /* Add customer’s code here */
        }
        break;

        /*!< Read request failed */
       case E_NTAG_EVENT_READ_FAIL:
        {
            /* Add customer’s code here */

        }
        break;

        /*!< Read request succeeded */
       case E_NTAG_EVENT_READ_OK:
        {
            /* Add customer’s code here */
        }
        break;

        /*!< Write request failed */
       case E_NTAG_EVENT_WRITE_FAIL:
        {
            /* Add customer’s code here */
        }
        break;

        /*!< Write request succeeded */
       case E_NTAG_EVENT_WRITE_OK:
        {
            /* Add customer’s code here */
        }
        break;

        /* Others ? */
        default:
        {
            /* Do nothing */
        }
        break;
    }
}
```

为了方便应用程序通过 NFC 方式交换 ZigBee 的 Network Key 网络密钥，NXP ZigBee3.0 SDK 还提供了 NTAG_NWK_eRead 和 NTAG_NWK_eWite 函数读、写 NTAG NWK NDEF 数据结构

tsNfcNwkPayload。如果使用这二个函数，则 NTAG_vTick()将被 NTAG_NWK_eTick()代替，通过返回值判断 NTAG NWK 是否读、写完成。

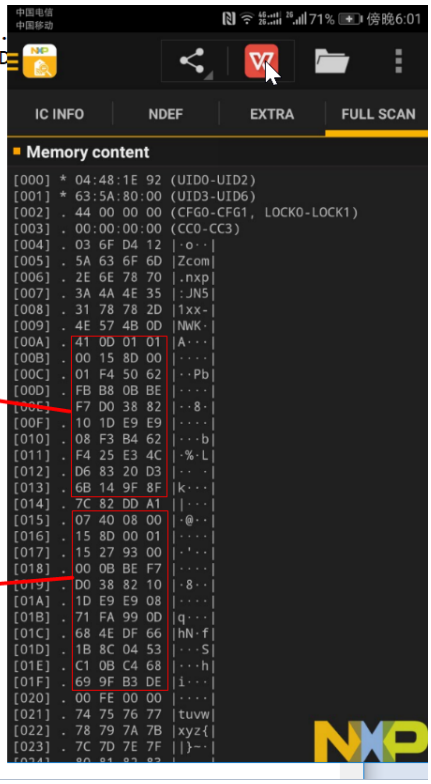
第一步，首次上电时设备生成随机的 Install Code 作为数据交换的 AES-128 密钥，调用函数 NTAG_NWK_eWite 把数据写入 NTAG 的 EEPROM 区域。下图右侧是通过手机 Apps 应用“NFC TagInfo”读取到的 NTAG EEPROM 内部数据。

```
0: APP_vNtagStart(x01) ICODE
0: ZTIMER_eStart(Ntag, 5) = 0.
5: APP_cbNtagEvent(0, -1, 0), eAppNtagState = ABSENT, eAppNtagMode = NWK.....
105: APP_cbNtagTimer(), NTAG_NWK_eTick() = 3 (READ_OK), u32Ntag = 32211, VALID
    sNfcNwkPayloadRead
        sNtag
            u8Version       = 13
            u8Command       = x41
            u8Sequence      = 13
            u16DeviceId     = x0101
            u64ExtAddress   = 00158d00:01f45062
            u16ShortAddress = xfbb8
            u8Channel       = 11
            u16PanId        = xbef7
            u64ExtPanId     = d0388210:1de9e908
            au8Key          = f3 b4 62 f4 25 e3 4c d6 83 20 d3 6b 14 9f 8f 7c
            u16Crc          = x82dd
        sNci
            u8Command       = x00
            u8Sequence      = 0
            u16DeviceId     = x0000
            u64ExtAddress   = 00000000:00000000
            u16ShortAddress = x0000
            u8Channel       = 0
            u16PanId        = x0000
            u64ExtPanId     = 00000000:00000000
            au8Key          = 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
            au8Mic          = 00 00 00 00
            u8KeySeqNum     = x00.
```



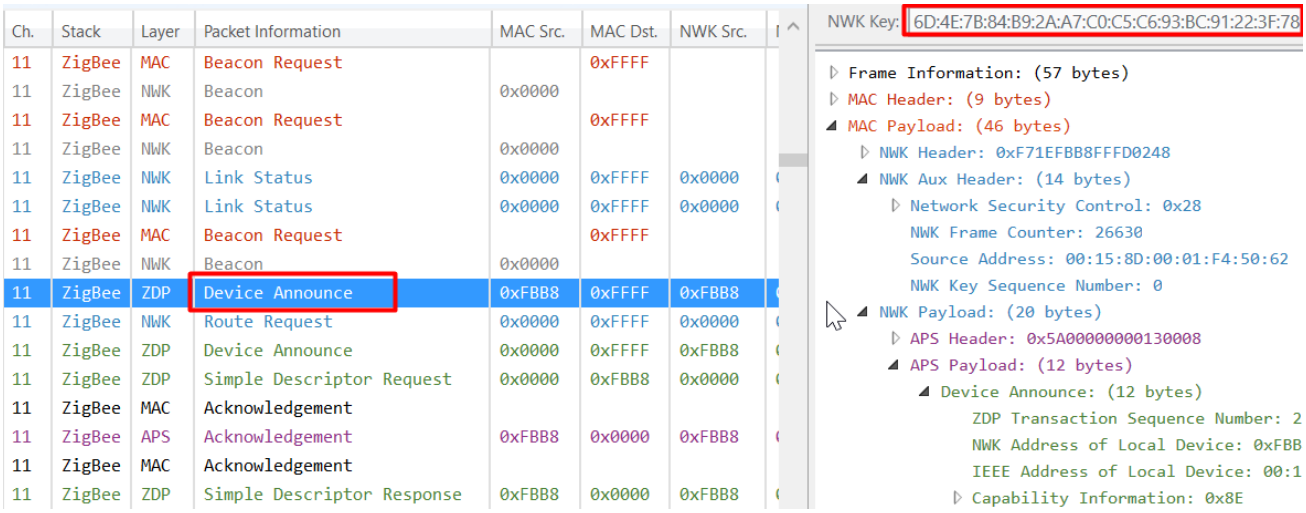
第二步，设备靠近 ZigBee Gateway 进行配对时，通过 PN7120 NFC Reader 读取上述数据。ZigBee Gateway 将 ZigBee Network Key 网络密钥使用 sNfcNwkPayloadWrite.sNtag.au8Key 作为密钥，进行 AES-128 加密后把密文写入 sNfcNwkPayloadRead.sNci 区域。

```
0: ZTIMER_eStart(Ntag, 5) = 0.
5: APP_cbNtagEvent(0, -1, 0), eAppNtagState = ABSENT, eAppNtagMode = NWK.....
105: APP_cbNtagTimer(), NTAG_NWK_eTick() = 3 (READ_OK), u32Ntag = 32211, VALID
    sNfcNwkPayloadRead
        sNtag
            u8Version       = 13
            u8Command       = x41
            u8Sequence      = 13
            u16DeviceId     = x0101
            u64ExtAddress   = 00158d00:01f45062
            u16ShortAddress = xfbb8
            u8Channel       = 11
            u16PanId        = xbef7
            u64ExtPanId     = d0388210:1de9e908
            au8Key          = f3 b4 62 f4 25 e3 4c d6 83 20 d3 6b 14 9f 8f 7c
            u16Crc          = x82dd
        sNci
            u8Command       = xa1
            u8Sequence      = 7
            u16DeviceId     = x4008
            u64ExtAddress   = 00158d00:01152793
            u16ShortAddress = x0000
            u8Channel       = 11
            u16PanId        = xbef7
            u64ExtPanId     = d0388210:1de9e908
            au8Key          = 71 fa 99 0d 68 4e df 66 1b 8c 04 53 c1 0b c4 68
            au8Mic          = 69 9f b3 de
            u8KeySeqNum     = x00
105: bNtagNciCmdJoinWithCode()
        GetKey()        = 1
        au8NetworkKey   = 6d 4e 7b 84 b9 2a a7 c0 c5 c6 93 bc 91 22 3f 78
        PDM_eSaveRecordData(PDM_ID_APP_NFC_NWK_NCI) = 0
```



第三步，设备重新上电复位后，通过 NTAG_NWK_eRead 读取 NTAG 的 EEPROM 数据放入 sNfcNwkPayloadRead 数据结构。以 sNfcNwkPayloadRead.sNtag.au8Key 为密钥，解密获得 ZigBee 的 Network Key 网络密钥 au8NetworkKey[16]。在 sNfcNwkPayloadRead.sNci 中还有 ZigBee Channel 信道，ZigBee PanId，网关命令 u8Command 等信息。其中 sNfcNwkPayloadRead.sNci.u8Command 指示入网/退网命令。如果是入网，调用 APP_bNtagNciCmdJoinWithCode()函数设置相应的协议栈 BDB 参数，实现设备 NFC 配对入网。

从下图可以看到，使用 NTAG I²C 可以通过轻触来实现快速简单的入网体验。不需要通过无线方式传输密钥，既简单快捷，又避免了网络密钥被窃取的风险。



ZigBee 网关 PN7120 NFC Reader。NXP PN71xx NFC 解决方案，完全兼容 NFC Forum 协议，并提供 Linux®、Android™ 和 WinIoT 平台的驱动，客户可以使用该解决方案进行最快速的设计。这些 NFC 控制器支持最流行的平台，包括 Raspberry Pi®、BeagleBone® Black 以及以 Arduino® 接口为特点的任意平台，比如 Kinetis®、LPCXpresso 和 i.MX 系列 MCU。

	PN7120	PN7150
嵌入式固件	是	是
NFC标签	Type 1, 2, 3, 4, 5	Type 1, 2, 3, 4, 5
RF驱动电压	2.7或3.3 V	3.0到4.75 V
读写器模式	MIFARE®, FeliCA, ISO/IEC 15693	MIFARE®, FeliCA, ISO/IEC 15693
点对点模式	ISO/IEC 18092 目标方与发起方（有源与无源）	ISO/IEC 18092 目标方与发起方（有源与无源）
卡模拟模式	NFC Forum T4T (ISO/IEC 14443 A和B)	NFC Forum T3T和T4T (FeliCa & ISO/IEC 14443 A 和 B)
NFC Forum兼容性	是	是
封装	VFBGA49	HVQFN40
负载调制方案	无源	有源

NXP 提供了运行在 OpenWRT Linux 平台上的 ZigBee 网关参考设计 JN-AN-1222-IoT-Gateway-Host-with-NFC 。在这个参考设计中包含了 PN7120 NFC 的底层驱动和 ZigBee Control-Bridge 的完整实现。用户可以通过 Web GUI 操作界面控制 ZigBee 网络。