获取 Long Term Key

当在 SDK 中启用了 bonding 功能后，协议栈会把配对过程产生的一些信息保存在 FLASH 的 NVM 区域。以下代码演示了将这些信息读出来的过程（其中就包含 LTK）：

```
#define NVM_BOND_INFO_ADDR        (0x1007F000u) // NVM 的起始地址，根据实际情况进行修改

#pragma pack(1) // 按字节对齐

typedef struct bond_info_s
{
    uint8_t nvm_record_header[4];
    uint8_t reserved_0[4];
    uint8_t ediv[2];
    uint8_t rand[8];
    uint8_t ltk[16];
    uint8_t irk[16];
    uint8_t csrk[16];
    uint8_t peer_addr[6];
    uint8_t reserved_1[12];
}bond_info_t;

#pragma pack() // 恢复默认的对齐方式

void print_bond_info(void)
{
    bond_info_t *p_bond_info = (bond_info_t *)NVM_BOND_INFO_ADDR;

    PRINTF("bond information: \r\n");

    PRINTF("ediv:     : ");
    for (uint8_t i = 0; i < 2; i++)
    {
        PRINTF("%02x ", p_bond_info->ediv[i]);
    }
    PRINTF("\r\n");


    PRINTF("rand      : ");
    for (uint8_t i = 0; i < 8; i++)
    {
        PRINTF("%02x ", p_bond_info->rand[i]);
    }
    PRINTF("\r\n");

    PRINTF("ltk       : ");
    for (uint8_t i = 0; i < 16; i++)
    {
        PRINTF("%02x ", p_bond_info->ltk[i]);
    }
    PRINTF("\r\n");

    PRINTF("irk       : ");
    for (uint8_t i = 0; i < 16; i++)
```

```
    {
        PRINTF("%02x ", p_bond_info->irk[i]);
    }
    PRINTF("\r\n");


    PRINTF("csrk      : ");
    for (uint8_t i = 0; i < 16; i++)
    {
        PRINTF("%02x ", p_bond_info->csrk[i]);
    }
    PRINTF("\r\n");

    PRINTF("peer addr : ");
    for (uint8_t i = 0; i < 6; i++)
    {
        PRINTF("%02x ", p_bond_info->peer_addr[i]);
    }
    PRINTF("\r\n");
}
```

以上 print_bond_info() 函数，可在配对成功后调用，如：

```
/* BLE Security v4.2 is supported: BLE stack FW version >= 2.x */
void aci_gap_pairing_complete_event(uint16_t Connection_Handle,
                                    uint8_t Status,
                                    uint8_t Reason)
{
    if (Status == BLE_STATUS_SUCCESS)
    {
        print_bond_info();
    }
}
```

> 注意。获取 LTK 的方式，上文已经提供了示例代码，用户只需要将示例代码引用到自己的工程里即可。对于 BLUENRG-LP 的其他一些工程，需要依据下文介绍的方法进行适配。
>
> 1. 在 SDK 中使能绑定功能，以使协议栈将配对信息保存在 FLASH 里（否则不会保存）：
>
>    ```
>    #define SLAVE_BONDING_USAGE      BONDIN
>    ```
>
> 2. 编译 SDK 的 security 工程，并打开 Keil 的 MAP 文件，查看 NVM 的起始地址：
>
>    ```
>    Execution Region REGION_FLASH_NVM (Exec base: 0x1007f000, Load base:
>    0x10060c1c, Size: 0x00001000, Max: 0x00001000, ABSOLUTE, UNINIT)
>    ```
>
>   可见：0x1007F000。
>
>   对于不同的 SDK 工程，此 NVM 的起始地址可能会不同，用户需要根据自己的工程、对上文提供的 获取 LTK 的示例代码进行修改，即修改以下宏
>
>    ```
>    #define NVM_BOND_INFO_ADDR        (0x1007F000u) // NVM 的起始地址，根据实际
>    情况进行修改
>    ```