LPC55S6x 1B 芯片 SB 文件的生成及使用

有些客户在用指令'receive-sb-file'烧写 SB 文件到 LPC55S6x 1B 版本芯片时会遇到 SB 文件加载错误的情况。和 LPC55S6x 0A 相比，1B 芯片的 Secure Boot 进行了较大的改动。为了解决这一问题，我们首先需要了解二者的不同之处。

1. SB2.1 vs. SB2.0

SB2 文件内容在 elftosb 的用户手册中进行了介绍。SB2 文件被加载后，芯片会执行 SB 文件中配置文件所包含的配置指令。被下载的 image 路径包含在“source”.bd 文件中。文本文件（*sbkek.txt*）中的 SB 密钥用于 elftosb 命令行工具的加密。

0A 版本的 LPC55S6x 芯片支持 SB2.0 格式。

1B 版本的 LPC55S6x 芯片支持 SB2.1 格式。

SB2.0 和 SB2.1 的主要区别在于数字签名的应用。

SB2.0 只有加密，数字签名为可选项；SB 2.1 是加密+数字签名模式，这里的数字签名为必选项。

2. SB2 文件的生成和使用

2.1 SB2.0 生成范例（只有加密）：

elftosb -f lpc55xx -k "sbkek.txt" -c "commandFile.bd" -o "output.sb2" "input.bin"

其中

-f = family lpc55xx(lpc55xx系列)

-k = KEK文件

-c = 命令文件

-o = 生成的SB2

Files...= 输入文件("*input.bin*"，通常是输入的plain image)，用来替换命令文件中定义源文件。bin文件也可以写死在命令文件（"*commandFile.bd*"）中，来取代*elftosb*中的文件输入项。

"commandFile.bd"内容：

```
options {
flags = 0x4; // 0x8 encrypted + signed, 0x4 encrypted
buildNumber = 0x1;
productVersion = "1.00.00";
componentVersion = "1.00.00";
}
sources {
inputFile = extern(0);
}
```

section (0) {
erase 0x0..0x40000;
load inputFile > 0x0;
}

2.2 SB2.1生成范例(加密+数字签名)

✓ 1个根密钥

*elftosb.exe -f lpc55xx -k "sbkek.txt" -c "commandFile.bd" -o "output.sb2" -s "selfsign_privatekey_rsa2048.pem" -S "selfsign_v3.der.crt" -R "selfsign_v3.der.crt" -h "RKTH.bin" "input.bin"*4

✓ 4 个根密钥

elftosb.exe -f lpc55xx -k "sbkek.txt" -c "commandFile.bd" -o "output.sb2" -s private_key_1_2048.pem -S certificate_1_2048.der.crt -R certificate_1_2048.der.crt -R certificate_2_2048.der.crt -R certificate_3_2048.der.crt -R certificate_4_2048.der.crt -h "RHKT.bin" "input.bin"

其中

-f = family lpc55xx(lpc55xx系列)

-k = KEK文件

-c = 命令文件

-o = 生成的SB2文件

-s = 用于签名的证书私钥

-S = 证书链中的证书，证书链中的每一个证书都必须按照证书创建的顺序使用-S 进行指定（根证书排第一位）

-R = 根证书，可以指定 1-4 个根证书，每个根证书必须使用新的-R 转换方式进行指定，其中一个根证书必须是使用-S 转换方式指定的第一个证书

-h= elftosb 生成的二进制输出文件，其中包含所有根证书（RKTH）的散列值，要上传至芯片对应的寄存器中

Files...= 输入文件("*input.bin*"，通常是输入的plain image)，用来替换命令文件中定义源文件。bin文件也可以写死在命令文件（"*commandFile.bd*"）中，来取代*elftosb*中的文件输入项。

"commandFile.bd"内容：

```
options {
flags = 0x8; // 0x8 encrypted + signed, 0x4 encrypted
buildNumber = 0x1;
productVersion = "1.00.00";
componentVersion = "1.00.00";
}
sources {
inputFile = extern(0);
}
section (0) {
    erase 0x0..0x40000;
```

```
load inputFile > 0x0;
}
```

2.3 SB2 文件使用下载

更新过的二进制文件所生成的SB2文件可以通过ISP指令集的“ receive-sb-file”指令将其加载到芯片Flash中。

```
blhost -p COMxx receive-sb-file <path to the secured binary(.sb2)>
```

注意：在烧写 SB2.1 文件前，RKTH 必须已写入到 CMPA 中（参见 AN12283，5.5 CMPA preparation），并且使能位于 CFPA 的 ROTKH_REVOKE（地址 0x9DE18）的 RoT 密钥（参见 AN12283 5.4 CFPA preparation）

成功加载 SB2 文件后，MCU 将按照 SB 配置文件（.bd 文件）中的配置执行该文件。 执行文件时，将擦除从 0x0 到 0x40000 的内部 flash 地址。完成 flash 擦除操作后，烧写文件将从地址 0x0 开始下载。

完成上述操作后重启设备，更新的 image 将加载到内部 flash 执行。