Preface

An incomplete framework (the flash programming code, the device's serial I/O code and the device's timer code has to be provided) for an obfuscating bootloader instead of an encrypting bootloader is presented in order not to infringe the US Export Control. This issue can be fixed easily by replacing the *primitives* (at least the hash function or the 128-bit block permutation) of the bootloader with secure code, and by omitting the writing of the obfuscation key into the firmware file (by commenting out line 18 of *makeboot* ➔ *Program.cs*; the obfuscated firmware file can be opened as a JSON file with `TSF2JSON.EXE`, see also https://www.tellert.de/?product=tsf).

File Structure

Firmware Obfuscation.pdf: Official documentation about the obfuscation algorithm and file format.

Firmware Obfuscation – Internals.pdf: Internals about the obfuscation algorithm.

appldr: Application loader with GUI which is used to transfer the obfuscated firmware file to the target device via a serial port.

appldrc: Application loader with CLI which is used to transfer the obfuscated firmware file to the target device via a serial port.

bootldr: Source code for the bootloader of a microcontroller to accept an obfuscated firmware file via a serial port.

makeboot: Command line program to generate both the modified bootloader hex file and the obfuscated firmware file. (A hex file can either be an Intel hex file or a Motorola S-record file.)

makecfg: Command line program to generate a configuration file, or to merge an obfuscated firmware file with a configuration.

A build of the programs requires Microsoft Visual Studio 2013 (or newer).

appldr, appldrc, makeboot and *makecfg* require the Microsoft .NET Framework 4 which is part of Windows 8 or newer, or which can be downloaded from https://www.microsoft.com/en-us/download/details.aspx?id=17718.
And the Microsoft .NET Framework 4 Full Language Pack can be downloaded from https://www.microsoft.com/de-de/download/details.aspx?id=3324 .

Generating the Individualized Bootloader Hex File

The individualized bootloader hex file is generated at the command line with

```
makeboot device-number
```

E. g. the command

```
makeboot 123456
```

will generate the bootloader hex file for the target device with serial number 123456. This file can be flashed with a standard flash application to the target microcontroller.

Generating the Obfuscated Firmware File

The obfuscated firmware file is generated at the command line with

```
makeboot src-file target-file
```

The *src-file* has to be of hex format, and the *target-file* will be the obfuscated firmware file (*.tsf).

Generating an Erase File

The file which erases the firmware from a device (to clear all ROM of the target device) is generated at the command line with

```
makeboot
```

Generating a Configuration File

The configuration file is generated at the command line with

```
makecfg [-d cfgData.bin] cfg.bin target.tsf
```

where the optional *cfgData.bin* is the binary configuration data file, *cfg.bin* is the binary configuration file, and *target.tsf* is the target file.

Merging an Obfuscated Firmware File with a Configuration File

The merging is generated at the command line with

```
makecfg firmware.tsf cfg.tsf firmwareCombinedWithCfg.tsf
```

where *firmware.tsf* is the obfuscated firmware file, *cfg.tsf* is the configuration file, and *firmwareCombinedWithCfg.tsf* is the target file which contains both firmware and configuration.

Bootloader

The bootloader needs to be modified for the target device. It compiles directly as a Win32-API CLI program, if it is left untouched. In this case, the compiled executable can be used, together with *appldr* or *appldrc*, to verify/debug the transfer of the obfuscated firmware file via a serial null-modem cable. Note that the *slower* serial port (e. g. a hardware COM port instead of a virtual COM port) should be assigned to the application loader.

The data type *application_info_t* (in *appinfo.h*) is for a microcontroller of up to 32-bit width, and it is limited to 256 bytes of code. It is also the last ROM image block of the firmware file.

The default settings are for a Renesas RX64M microcontroller. The adjustable settings are at the beginning of *bootldr.h* (project *bootldr*) and of *Program.cs* (project *makeboot*).

Default Memory Map of the Target Device

`0x00100000 – 0x0010ffff`	*configData* area (data flash memory)
`0x00120050 - 0x0012005f`	Flash memory access ID code
`0xffc00000`	Begin of flash memory
`0xffc00000 - 0xffff4fff`	Application area
`0xffff5f00 - 0xffff5fff`	*appInfo* (at a block end)
`0xffff6000 - 0xffff7eff`	(intentionally left unused to the extend of the *configInfo*

	block)
`0xffff7f00 - 0xffff7fff`	*configInfo* (at a block end)
`0xffff8000 - 0xffffdff`	Bootloader area
`0xffffe00 - 0xffffeff`	*hwInfo*
`0xfffffffc - 0xffffffff`	Application entry point

Entries of *appldr.xml*

autoClose	If true, automatically terminate *appldr.exe* when finished
firmwareName	Name of the firmware
maxBaudRate	Maximal baud rate which should be used (Either 0, 9600, 57600, 115200, or 230400)
port	Optional name of the serial port (e. g. "COM1")
regName	Name of the registry key (e. g. "DEVICE")
startPage	Number of the start page (default: 0)
timeoutT0	Timeout in msec for the first block incl. erasing (default: 5 s)
timeoutT1	Timeout in msec for the following blocks (default: 1 s)
title	Title of the application
tsfFileName	Name of the obfuscated firmware file

Entries of *makeboot.xml*

appEntryAddress	Address of the application entry point
appInfoAddress	Start address of *appInfo*
build	Build number part of the application version
config	Configuration number part of the hardware version
configInfoAddress	Start address of *configInfo*
customer	Customer number (hardware info)
customerIdMax	Maximal value for *customer*
customerIdMin	Minimal value for *customer*
dataInfoAddress	Start address of *dataInfo*
dateMax	Maximal value for date
dateMin	Minimal value for date
deviceBootLoaderHexFile	Name of the destination bootloader hex file
deviceIdCodeAddress	Start adress of flash ROM access ID code
deviceNumberMax	Maximal value for *deviceNumber*
deviceNumberMin	Minimal value for *deviceNumber*
eraseTsfFile	Target name of the TSF file (for erasing)

eraseBegin	Start address of the erasable region
eraseDataBegin	Start address of the erasable data region
eraseDataSize	Size of the eraseable data region
eraseRegions	Flags of the erasable regions
eraseSize	Size of the eraseable region
features	Application features number
hardwareFeatures	Hardware features
hardwareFeaturesReq	Requested hardware features
hardwareFeaturesXorMask	XOR mask for hardware features
hardwareId	Hardware ID number
hardwareIdMin	Minimal value of *hardwareId*
hardwareIdMax	Maximal value of *hardwareId*
hardwareVersion	Hardware version number
hardwareVersionMin	Minimal value of *hardwareVersion*
hardwareVersionMax	Maximal value of *hardwareVersion*
hwInfoAddress	Start address of *hwInfo*
id	Application ID number
internalDeviceNumber	Internal device number
internalDeviceNumberMax	Maximal value of *internalDeviceNumber*
internalDeviceNumberMin	Minimal value of *internalDeviceNumber*
manufacturerId	Id of the manufacturer (reserved values: 1)
manufacturerIdMax	Maximal value of the *manufacturerId*
manufacturerIdMin	Minimal value of the *manufacturerId*
maxTsfSize	Maximal file size of the TSF file
nmiIsrAddress	Entry point for the NMI ISR
orgBootLoaderHexFile	Source name of the bootloader hex file
ownDrmBlock	If *true,* the DRM block is separated from the data blocks
programBegin	Start address of the programmable configuration area
programDataBegin	Start address of the programmable configuration data area
programDataSize	Size of the programmable configuration data area
programSize	Size of the programmable configuration area
romBegin	Start address of the flash memory
strDesc	Device description string
strKey	Master key
strName	Device name string
strVersion	Device version string
tinyBootloader	If *true,* the tiny bootloader format (instead of the classic

	bootloader format) is used
utcFile	File name of an incremented value for uniqueness
versionMajor	Major version number of the application
versionMinor	Minor version number of the application
versionRevision	Revision number of the application

Entries of *makecfg.xml*

buildMax	Maximal value for the application build
buildMin	Minimal value for the application build
configStart	Start address of the configuration area
configDataStart	Start address of the configuration data area
configMax	Maximal value for the application config
configMin	Minimal value for the application config
customerIdMax	Maximal value for *customer*
customerIdMin	Minimal value for *customer*
dateMax	Maximal value for date
dateMin	Minimal value for date
deviceNumberMax	Maximal value for *deviceNumber*
deviceNumberMin	Minimal value for *deviceNumber*
eraseRegions	Regions to be erased
featuresReq	Request application features
featuresXorMask	XOR mask for application features
hardwareFeaturesReq	Requested hardware features
hardwareFeaturesXorMask	XOR mask for hardware features
hardwareIdMin	Minimal value of *hardwareId*
hardwareIdMax	Maximal value of *hardwareId*
hardwareVersionMin	Minimal value of *hardwareVersion*
hardwareVersionMax	Maximal value of *hardwareVersion*
idMax	Maximal value of the application id
idMin	Minimal value of the application id
internalDeviceNumberMax	Maximal value of *internalDeviceNumber*
internalDeviceNumberMin	Minimal value of *internalDeviceNumber*
manufacturerIdMax	Maximal value of the *manufacturerId*
manufacturerIdMin	Minimal value of the *manufacturerId*
manufacturingDateMax	Maximal value of the *manufacturingDate*
manufacturingDateMin	Minimal value of the *manufacturingDate*
versionMax	Maximal value of the application version

versionMin	Minimal value of the application version

Implementation Limitations

Eraseable areas are not implemented (*erase regions* can be used instead). The bootloader files are assumed to have at least one DRM item of type *hardwareId*, otherwise the file will not be accepted by the bootloader. Programmable areas are expected to begin at addresses with low byte equals zero. The size of programmable areas is expected to be an integer multiple of 256. *Firmware.GetConfigBlocks*() should be called after *Bootloader.ReadVersion*() or *Bootloader.ReadInfo*() is executed because otherwise the block format is unknown.

Internet

The homepage of the firmware obfuscation can be found at:

https://www.tellert.de/?product=fo