U-Boot: Verified RSA

Boot on ARM target

Jagannadh Satradharudu Teki
U-Boot Mini Summit- Edinburgh, 2013 Oct
Agenda

- Zynq U-Boot
- SPI Custodianship
- Verified Boot
- RSA Concept
- U-boot Verified RSA Boot
- Current u-boot state(Simon’s support)
- U-boot needs
- Demo run
- TODO
- References
Good customer support till now – feature additions SPI/QSPI, support new boards, d-caches and bug fixes

~75% of u-boot-xlnx code is in ML, rest will push soon.
SPI Custodianship

[u-boot.git] / doc / SPI / status.txt

1 Status on SPI subsystem:
2 ==========================
3
4 SPI COMMAND (common/cmd_sf, cmd_spi):
5 -
6
7 SPI FLASH (drivers/mtd/spi):
8  - sf_probe.c: SPI flash probing code.
9  - sf_ops.c: SPI flash operations code.
10 - sf.c: SPI flash interface, which interacts controller driver.
11 - Bank Address Register (Accessing flashes > 16Mbytes in 3-byte addressing)
12 - Added memory_mapped support for read operations.
13 - Common probe support for all supported flash vendors except, ramtron.
14
15 SPI DRIVERS (drivers/spi):
16 -
17
18 TODO:
19  - Runtime detection of spi_flash params, SFDP(if possible)
20  - Add support for multibus build/accessing.
21  - Extended read commands support(dual read, dual IO read)
22  - Quad Page Program support.
23  - Quad Read support(quad fast read, quad IO read)
24  - Dual flash connection topology support(accessing two spi flash memories with single cs)
25  - Banking support on dual flash connection topology.
26  - Need proper cleanups on spi_flash and drivers.
27
Verified Boot

- Verified – Secure – Trusted boot
- Verify the loaded software to ensure that it is authorized during boot.
- Benefits:
  - Prevent from malware
  - Provide authorized read access
  - Machine safe – runs only signed software
  - Possible to file-upgrade the software
RSA Concept

**Signing**

- openssl
- .key .crt
  - Public Key
  - Private Key
- Signer
- software image

**Verification**

- Trusted source
  - Public Key
- Verifier
  - signed_img
  - Yes/No
U-boot Verified RSA Boot

**Signing**
- openssl
- .key .crt
- Private Key

**Verification**
- u-boot dts
- Public Key

**mkimage**
- fit input
- software image
- signed_img

**U-boot Verification**
- OK/Bad Data Hash

Note: We can also use mkimage to write pubkey on dtb
<table>
<thead>
<tr>
<th>Date</th>
<th>Author</th>
<th>Description</th>
<th>Git Commit</th>
<th>Git Branch</th>
<th>Git Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013-06-26</td>
<td>Dirk Behme</td>
<td>spil: mxc_spi: Fix pre and post divider calculation</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>Add verified boot information and test</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>sandbox: config: Enable FIT signatures with RSA</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>image: Add support for signing of FIT configurations</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>libfst: Add fdt_find_regions()</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>mkmage: Add -r option to specify keys that must be...</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>mkmage: Add -c option to specify a comment for key...</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>mkmage: Add -f option to modify an existing .fit file</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>mkmage: Add -k to write public keys to an FDT blob</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>mkmage: Add -k option to specify key directory</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>image: Add RSA support for image signing</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>image: Support signing of Images</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>image: Add signing infrastructure</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>x86: config: Add tracing options</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>x86: Support tracing function</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>exynos: config: Add tracing options</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>exynos: Avoid function instrumentation for microsecond</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>arm: Implement the ‘take’ go command</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>Refactor the bootm command to reduce code duplication</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>Exynos boot OS arguments</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>Add a simple test for sandbox trace</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>sandbox: Support trace feature</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>Add etootl to decode profile data</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>Add trace support to generic board</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>Support tracing in config.mk when enabled</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>Add a trace command</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>Add trace library</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>Add function to print a number with grouped digits</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>bloadmsg: Correct print types</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>Show stdio on error in fit-test</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>2013-06-26</td>
<td>Simon Glass</td>
<td>Fix missing return in do_mem_loop()</td>
<td>commit</td>
<td>main</td>
<td></td>
</tr>
</tbody>
</table>
U-boot needs

- **Enable FIT**
  - `CONFIG_FIT` - enable support for the FIT uImage format
- **Enable FDT**
  - `CONFIG_OF_CONTROL`
  - `CONFIG_OF_SEPARATE`
- **Enable verified boot**
  - `CONFIG_FIT_SIGNATURE` - enables signature verification of FIT images
  - `CONFIG_RSA` - enables the RSA algorithm used for FIT image verification
Demo run...

- Build FDT u-boot
- Build rsa_signed image
- Build FDT u-boot with public key
- Run rsa_signed image
Build FDT u-boot

Setup the toolchain:
http://www.wiki.xilinx.com/Zynq+Base+TRD+14.5#x-5 Building the U-boot Boot Loader

Clone u-boot-spi.git
$ git clone git://git.denx.de/u-boot-spi.git
$ cd u-boot-spi
$ git checkout -b master-xlnx origin/master-xlnx

U-boot build
$ make zynq_zed_config
$ make DEVICE_TREE=zynq-zed -j4
```c
/dts-v1/
/
{  
description = "Simple image with single Linux kernel, FDT blob and ramdisk";
#address-cells = <3x1>;
images {
  kernel@1 {
    description = "Zynq Linux kernel";
    data = /incbin("./vmlinux.bin.gz");
    type = "kernel";
    arch = "arm";
    os = "linux";
    compression = "gzip";
    load = <0x800000>;
    entry = <0x800000>;
    hash@1 {
      algo = "sha1";
    };
    signature@1 {
      algo = "sha1,rsa2048";
      key-name-hint = "dev";
    };
  };

  fdt@1 {
    description = "ZED board Flattened Device Tree blob";
    data = /incbin("./devicetree.dtb");
    type = "flat dt";
    arch = "arm";
    compression = "none";
    hash@1 {
      algo = "sha1";
    };
    signature@1 {
      algo = "sha1,rsa2048";
      key-name-hint = "dev";
    };
  };

  ramdisk@1 {
    description = "Ramdisk Image";
    data = /incbin("./ramdisk.image.gz");
    type = "ramdisk";
    arch = "arm";
    os = "linux";
    compression = "gzip";
    load = <0x80000000>;
    entry = <0x80000000>;
    hash@1 {
      algo = "sha1";
    };
    signature@1 {
      algo = "sha1,rsa2048";
      key-name-hint = "dev";
    };
  };
}
}
configurations {
  default = "conf@1";

  conf@1 {
    description = "Boot Linux kernel, FDT blob and ramdisk";
    kernel = "kernel@1";
    fdt = "fdt@1";
    ramdisk = "ramdisk@1";
  };
};
```
Build rsa_signed

- RSA key generation:
  - Create RSA key pair
    
    ```
    openssl genrsa -F4 -out mykeys/dev.key 2048
    ```
  - Create a certificate contains public key
    
    ```
    openssl req -batch -new -x509 -key mykeys/dev.key -out mykeys/dev.crt
    ```

- Create dtb for existing u-boot dts
  
  ```
  dtc -p 0x1000 board/xilinx/dts/zynq-zed.dts -O dtb -o zynq-zed.dtb
  ```
  
  ```
  cp zynq-zed.dtb zynq-zed-pubkey.dtb
  ```

- Sign the images with mykeys
  
  ```
  DTC_OPS="-I dts -O dtb -p 2000"
  ```
  
  ```
  mkimage -D "${DTC_OPS}" -f rsa.its -K zynq-zed-pubkey.dtb -k mykeys -r rsa_signed.img
  ```
Build FDT u-boot with public key

- For building FDT u-boot with public key externally
  $ make DEV_TREE_BIN=./zynq-zed-pubkey.dtb

u-boot-dtb.bin -> Is final FDT u-boot image with public key on it, hence the pubkey will used in verification process.
zyq-uboot> boot 0x2000000

## Loading kernel from FIT Image at 02000000 ...

Using 'conf@1' configuration
Verifying Hash Integrity ... OK
Trying 'kernel@1' kernel subimage
  Description: Zynq Linux kernel
  Type: Kernel Image
  Compression: gzip compressed
  Data Start: 0x800000f0
  Data Size: 2972178 Bytes = 2.8 MiB
  Architecture: ARM
  OS: Linux
  Load Address: 0x80000000
  Entry Point: 0x80000000
  Hash algo: md5
  Hash value: 3601aecd79bd62a71a43e72880a41d24
  Hash algo: sha1
  Hash value: 5c18a3632e839393499ea6d43e9fa861d5193
  Sign algo: sha1, rsa2048:dev
  Sign value: 1b63d3e6c027783626779f8fa4bebaed46d97d4d3ce4ce4e39f10aff4e79da2a796c84619806e6a8d7ae17
5d67f8a3l3f21378a84a6a0a2f7c7cc74d66ee9c7a6619b7a6365888bc8cffe73dc8b155dc5db252621c9b582e4df2cf95315c701dc53
35e56f93e566d0eb5c7b334aecc3e1375d45c5d9b9c2084683420378a0f9c34bbab724256e9fac56c9ab3375e0c8cd9334a6
4ed35f215b1386ae083e73809261c8e159d2a8a66c9b56d8f7447e1f183dd2542231579f4e69456d398d09ab50bed8e8f6
3369426c80ab41be2aad89df5918a5a8882dacc5a21313f48b22f54376d11ff4229c9587bedd99c7cc2bf448237b72372c5793194c56c372d

## Loading ramsdisk from FIT Image at 02000000 ...

Using 'conf@1' configuration
Verifying Hash Integrity ... sha1, rsa2048:dev+ md5+ sha1+ OK
## TODO

- Possible TODO’s @ doc/uImage.FIT/signature.txt
- **Signed_image creations support for bootable images (SPL) or FIT support in SPL ???**

```c
images {
    spl@1 {
        description = "Zynq SPL";
        data = /incbin("./SPL.bin");
        type = "spl";
        arch = "arm";
        compression = "none";
        load = <0x0>
        entry = <0x0>
        hash@2 {
            algo = "sha1";
        }
        signature@1 {
            algo = "sha1,rsa2048";
            key-name-hint = "dev"
        }
    }
    u-boot@1 {
        description = "Zynq u-boot";
        data = /incbin("./u-boot.bin");
        type = "u-boot";
        arch = "arm";
        compression = "none";
        load = <0x40000000>
        entry = <0x40000000>
        hash@2 {
            algo = "sha1";
        }
        signature@1 {
            algo = "sha1,rsa2048";
            key-name-hint = "dev"
        }
    }
}
```
References

➤ Zynq u-boot-xlnx.git repo
https://github.com/Xilinx/u-boot-xlnx

➤ For verified boot: doc/uImage.FIT/verified-boot.txt

➤ For signature: doc/uImage.FIT/signature.txt

➤ Sample sign its: doc/uImage.FIT/sign-configs.its

➤ Code for this demo run
http://git.denx.de/?p=u-boot/u-boot-spi.git;a=shortlog;h=refs/heads/master-xlnx

➤ Possible TODO’s on doc/uImage.FIT/signature.txt

➤ Any questions - mail to sjg@chromium.org CC u-boot@lists.denx.de, jagannadh.teki@gmail.com