XAPP1309 (v1.0) March 7, 2017

Measured Boot of Zynq-7000 All Programmable SoCs

Author: Lester Sanders

Summary

The secure boot functionality for the Zynq®-7000 All Programmable (AP) SoC provides the capability to authenticate all partitions loaded at boot using RSA-2048 authentication. It also supports advanced encryption standard (AES) encryption of partitions that need confidentiality. The Zynq-7000 AP SoC immutable BootROM includes security functions to provide a hardware root of trust (HROT) to protect against early load attacks.

This application note discusses a method to add measured boot capability to Zynq-7000 AP SoCs used in a connected environment. A server provides remote attestation that the embedded systems boot with trusted software over a secure network. The method uses a trusted platform module (TPM) to enhance the HROT functionality. The TPM provides cryptographic functions in a cost-effective, tamper-resistant device which are an effective complement to Zynq-7000 SoC security functions.

Download the reference design files for this application note from the Xilinx website.

Introduction

In most current applications, Xilinx FPGAs and SoCs are programmed once at the factory and often not reconfigured for the life cycle of the device. A method to add functionality and/or reduce the total cost of ownership (TCO) of an embedded system is to support field updates. In Zynq-7000 AP SoCs, the SoC and programmable logic can be updated, so field updates can be very effective. Field updates are typically done over the Internet, which opens up attacks on an embedded system to anyone with network access. Measured boot and network security are critical in firmware updates.

Figure 1 shows an example system environment that uses measured boot. A server manages the software load, update, and validation of fielded, embedded systems based on the Zynq-7000 AP SoC. The embedded systems connect to the server using Ethernet. In addition to updating software on the embedded systems, the server verifies that the correct, trusted software is loaded. This verification by the server, done at boot and run time, is remote attestation.

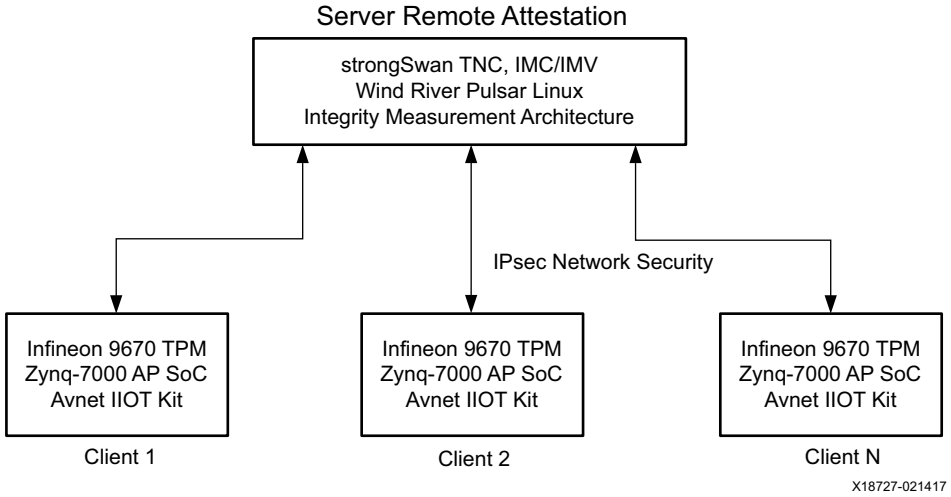


Figure 1: **Measured Boot of Zynq-7000 AP SoC Embedded Systems**

Remote attestation capability has been in Linux starting with 2.6.3, and is generally known as integrity measurement architecture (IMA). The Linux extended verification module (EVM) is used in conjunction with IMA. The term *measured boot* is used because the client returns a value, typically a secure hash algorithm-1 (SHA-1) digest, of each of the partitions loaded. In remote attestation, the server compares the measured logs with known good measurements.

The attestation server knows the characteristics (measurements) of the partitions loaded on the embedded systems, including partition size and digests. At load, the embedded systems send log files to the server containing partition measurements. The server verifies the measurement, and if a client loads software that is different than what is expected, the server executes a policy set up by the server administrator. A policy is a set of actions taken by the server based on measurement results. Policies in the reference design include *Allowed*, *Quaranteed*, *Blocked*, and *Isolated*.

An example of a policy is to keep the embedded system off the network, update the software, re-run remote attestation, and allow the client to connect to the network if the software can be trusted. Isolating a corrupted embedded system from the network limits its ability to corrupt other embedded systems. This is a typical policy of a server in remote attestation, not the only policy, because the policy is generally defined by the application.

Measured boot is done in addition to, not in place of, secure boot. Measured boot does not prevent malicious software from being loaded. The TPM enhances the HROT and increases the security of the software load/update process. The TPM is placed on the same board as the Zynq-7000 AP SoC. A device ID is associated with the Zynq-7000 SoC-TPM platform. The TPM provides cryptographic functions used in measure boot. The HROT is enhanced with the TPM because an adversary has to defeat both the Zynq-7000 AP SoC and the tamper-resistant TPM for a successful attack.

Figure 2 shows functional components of the Zynq-7000 AP SoC and the Infineon OPTIGA SLB 9670 TPM on the client platform.

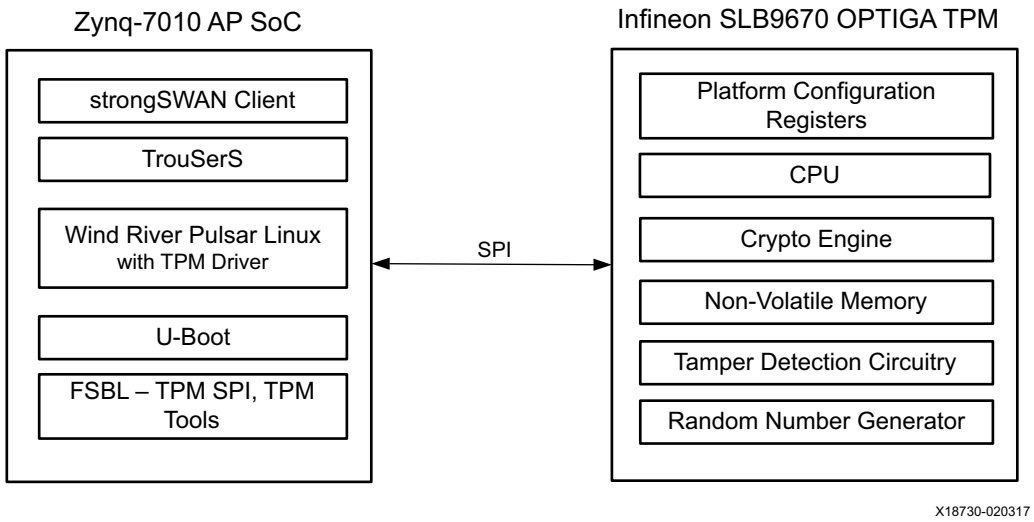


Figure 2: **Functional Diagram of Client Platform Based on Zynq-7000 AP SoC**

At power-up, the Zynq-7000 AP SoC on-chip BootROM code loads the first stage boot loader (FSBL). The FSBL loads U-boot, and U-Boot loads the Linux kernel, root file system, device tree, and Linux application software. In one approach to booting with a chain of trust, the BootROM authenticates/measures the FSBL, the FSBL authenticates/measures U-Boot, and U-Boot authenticates/measures the Linux partitions.

The SHA-1 measurement logs are stored in the TPM platform configuration registers (PCRs). Measurements of the BootROM and the FSBL are done by the FSBL and placed in the PCRs using a serial peripheral interface (SPI) connection. The measurements are transmitted to the server for remote attestation. The TPM cryptographically signs the SHA-1 values in PCRs so that partition measurements are not transmitted from the embedded system in plain text.

For remote attestation of firmware updates, the network connection between the attestation server and clients must be secure. IPsec functionality, including a privacy certificate authority (CA) that generates X.509 certificates, implements the transport layer security (TLS) handshake between the server and client(s). The network security used in the measured boot reference design is discussed in Network Security in Measured Boot, page 12.

Hardware and Software Requirements

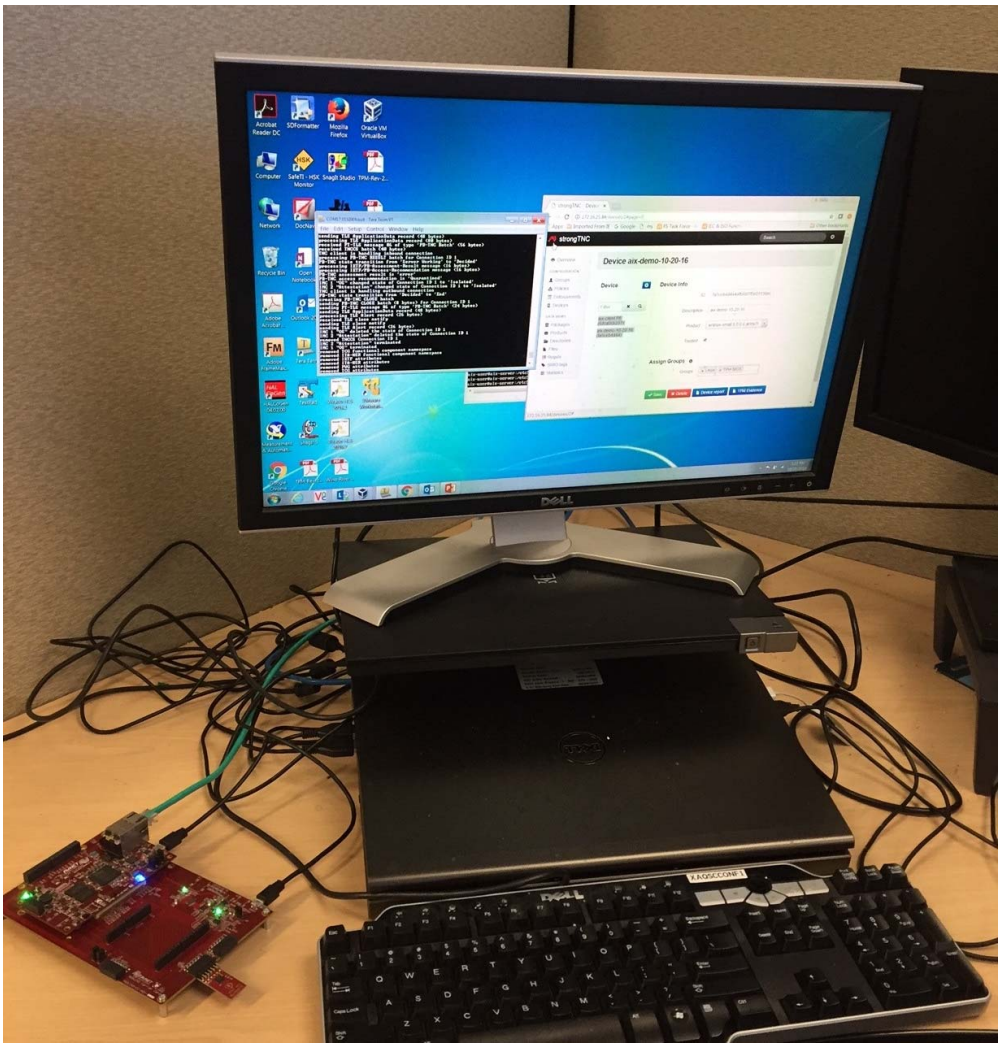
The hardware and software requirements for the reference system include the following:

- Avnet Industrial Internet of Things (IIoT) starter kit with MicroZed board
- Infineon OPTIGA TPM 1.2 SLB 9670 peripheral module (Pmod) compatible board
- Two USB type-A to USB mini-B cables (for UART and JTAG communication)
- Micro Secure Digital (microSD) memory card (16 GB)
- Ethernet cable
- Xilinx Software Development Kit 2017.1
- Xilinx Vivado® Design Suite 2017.1 (optional)
- Wind River Pulsar Linux 8.0
- VirtualBox 5.0.26 or higher (or VMware equivalent)
- Ubuntu or Ubuntu command line virtual machine image in Open Virtual Appliance format
- strongSwan TNC software
- Tera Term or equivalent communication terminal software

Depending on the system setup, all of the software listed above might not be necessary.

Reference System Description

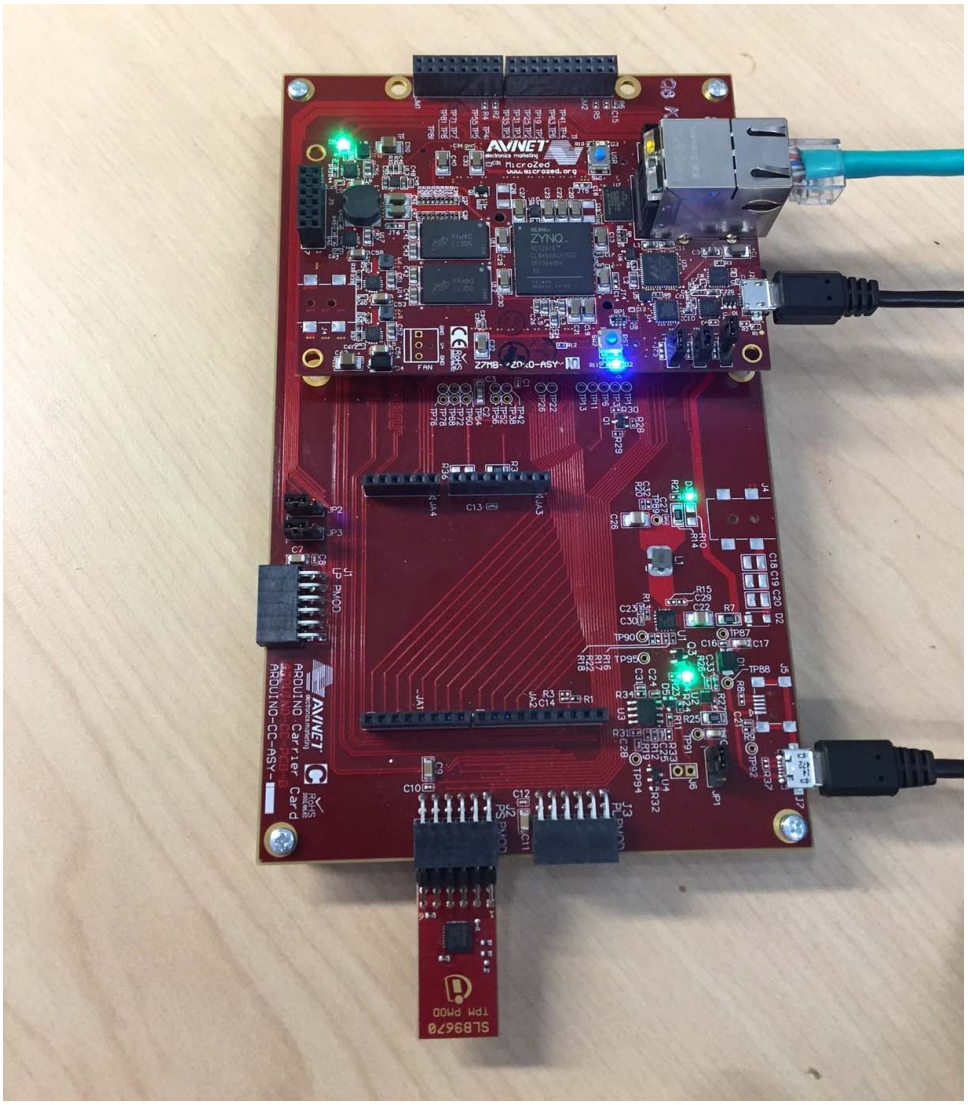
Figure 3 shows a desktop setup for the single client system used in the reference design.The client in the Avnet IIoT drives a communication terminal. The strongSwan attestation server runs from VirtualBox. A browser is used to view the measurements and the implementation of the policy.



X18725-020317

Figure 3: **Measured Boot Reference Design**

Figure 4 shows the Avnet IIoT starter kit with the MicroZed board mounted on the Avnet Arduino carrier card. It also shows the Infineon OPTIGA TPM 1.2 SLB 9670 Pmod plugged into the J2 PS PMOD connector.



X18724-020317

Figure 4: **Avnet IIOT Starter Board with Infineon OPTIGA SLB 9670 TPM Connection**

The Infineon OPTIGA SLB 9760 attached to PMOD connector J2 communicates to the Zynq-7000 AP SoC using the processing system (PS) serial peripheral interface (SPI) driver. Wind River Pulsar Linux (WRPL) 8.0 runs on MicroZed and includes the strongSwan client software. Prior to booting WRPL, the Zynq-7000 AP SoC runs the FSBL. The FSBL runs pre-boot authentication on the BootROM and FSBL.

The FSBL then executes PCR extend commands to log BootROM and FSBL SHA-1 digests into the PCR[0] and PCR[4]. The PCR extend command is similar to a write command. It differs from the write value in that the value loaded into the PCR is an SHA of all cumulative SHA values loaded into the PCR. Because a PCR extend rather than a write is done, an adversary is unable to read PCRs and get “expected” values.

Figure 5 shows three possible system setups. The strongSwan software running on an Ubuntu server does both the remote attestation and the network security. The Ethernet connection can be either direct from the PC to the Avnet board or dynamic host configuration protocol (DHCP) using Ethernet wall sockets. In (a), the server is run on an Ubuntu installation on either VirtualBox or VMware on the PC. In (b), an Ubuntu-based PC runs the strongSwan server. In (c), the server runs on an Amazon web server (AWS). Xilinx does not provide the AWS account.

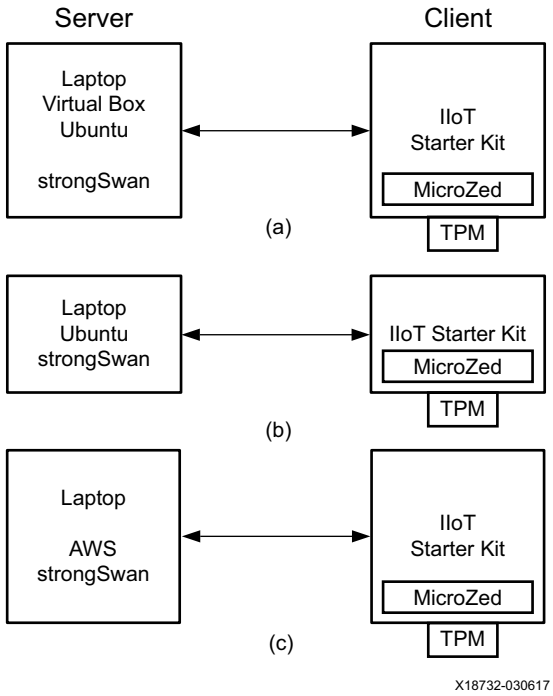


Figure 5: **Reference System Hardware Setup Options**

The reference system provides methods for a measured boot of a single client, providing remote attestation of early load software (BootROM, FSBL) which is new functionality for embedded systems. An actual connected system has multiple embedded devices, and the strongSwan server measures all Linux partitions loaded, not just the early load software.

Hardware Root of Trust

In Zynq-7000 AP SoCs, the HROT is based on the first code executed by the ARM® CPU0 at power-on. The code is stored in on-chip, metal-masked ROM, and is referred to as BootROM code. BootROM code is immutable, and its principle function is to perform device initialization and load the FSBL into read/writable on-chip memory (OCM). Neither the BootROM nor the OCM are accessible at device pins. The BootROM Configuration Flowchart figure in *Zynq-7000 All Programmable SoC Technical Reference Manual* (UG585) [Ref 1] provides the flow of the BootROM code functionality. If secure boot is specified, the BootROM authenticates the FSBL using the RSA-2048 standard prior to execution of the FSBL. The Zynq-7000 AP SoC HROT is enhanced by adding a TPM to the embedded platform. The TPM provides partition measurements, cryptographic functions, and secure key storage for keys used by the Zynq-7000 AP SoC.

In Zynq-7000 AP SoCs, the term *secure boot* is used to define the secure loading of the bitstream and software at power-on. The bitstream is loaded into on-chip configuration memory. Software partitions encrypted in non-volatile memory (NVM) are generally authenticated, decrypted, and copied to DDR memory. As shown in Figure 6, the RSA-2048 authentication uses a chain of trust sequentially on each partition loaded. Each partition can use its own private/public key pair. Partitions can optionally be authenticated using keyed-hash message authentication code (HMAC). In its simplest form, all partitions loaded by the FSBL are authenticated using RSA-2048 from the XILRSA library. In an alternative boot flow, U-Boot has access to the XILRSA library. In secure boot, if either RSA or HMAC authentication fails, the Zynq-7000 AP SoC transitions to a lockdown state.

A malicious actor needs to steal the RSA private key, which is not stored in the device, and the HMAC key. To protect against an insider attack, the SDK Bootgen key management can be split so that independent parties handle RSA, AES, and HMAC keys.

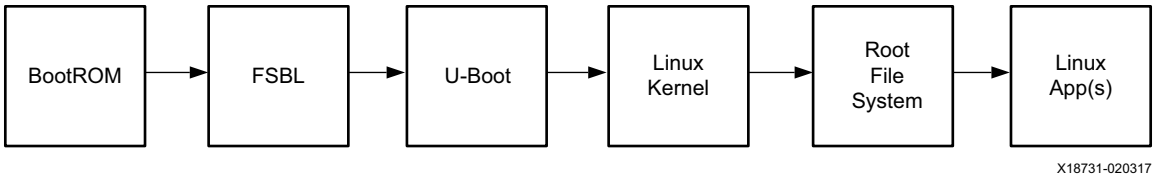


Figure 6: **Zynq-7000 AP SoC Chain of Trust Boot**

Measured Boot

Measured boot is recommended when embedded systems are connected to a network. Secure boot should still be used in systems which use measured boot. Secure boot and measured boot functionality are complementary. Connecting embedded systems to a network provides a method for firmware updates. Embedded systems connected to a network have a wider attack surface than closed systems. Hackers with network access are a common security threat. Remote attestation addresses this vulnerability during boot and run time. In secure and measured boot, all files/partitions are authenticated and measured.

The basics of measured boot are discussed in Integrity Measurement Architecture. The measured boot in the reference design uses a TPM for added security. Trusted Platform Module, Zynq 7000 SoC-TPM Interface, and Network Security in Measured Boot discuss measured boot using a TPM. Secure boot and measured boot do not use programmable logic resources, so the Zynq-7000 AP SoC unit cost is not affected when measured boot is used.

Using measurements, an attestation server can periodically execute run-time integrity checks on clients, and execute a policy based on the results. This is important in connected systems because the probability of an attack during run time is high. The ability to implement a policy is an improvement over the runtime integrity checker (RTIC) described in *Run Time Integrity and Authentication Check of Zynq-7000 AP SoC System Memory* (XAPP1225) [Ref 2].

Integrity Measurement Architecture

IMA provides the basis for measured boot. An overview of IMA is provided in strongSwan documentation [Ref 3].

In remote attestation with IMA, the server compares measurements received from clients with reference integrity measurements (RIMs) and acts according to a predefined policy. In the reference design, this is referred to as the policy decision point (PDP). After running measured boot, a server website provides a summary of measurements and policy for the reference design.

In an IMA implementation, the client runs integrity measurement collection (IMC). The server runs integrity measurement verification (IMV). Figure 7 shows an example log when all Linux partitions are measured.

```
ls /sys/kernel/security/ima
ascii_runtime_measurements  binary_runtime_measurements  runtime_measurements_count  violations

sudo cat /sys/kernel/security/ima/runtime_measurements_count
1271

sudo less /sys/kernel/security/ima/ascii_runtime_measurements
10 ef2be9c304d9bbbd8ecb40f0d296176d2b5d3078 ima-ng sha1:4663ed64e5dbbb9755a0914b1a15fa76a1797806 boot_aggregate
10 ef411bae164fd624ea94fc9ef82f892c82d78dcd ima-ng sha1:bbe98e20b850f3907611fb96354b5e007a9179f4 /init
10 bd32e452e14f84eb22d6ac9e9e1c61eeac3cd7a4 ima-ng sha1:dc3e621c72cde19593c42a7703e143fd3dad5320 /bin/sh
10 eefd4a6bebd6b001ff587c2335a3dd03535d5a17 ima-ng sha1:d11ce2e31ab441be705df3061a3d6fb7e41a504e /lib64/ld-linux-x86-64.so.2
10 8e8844cba6dc9df17c6980122890487f818e4b28 ima-ng sha1:34efdbd6d562ac04f7e02195022c3f65f7553bd2 /etc/ld.so.cache
10 1f60da15c941fe25a18ee4e8378f0bf3b447a0ab ima-ng sha1:65228a2bbff8ca52d2040ac55499b348f648cc81 /lib/x86_64-linux-gnu/libc.so.6
10 223eb68bfb9f72922506747d3bc4dd76d813b5da ima-ng sha1:65030975e1f3887efd00fbb568f00409b7c256d0 /conf/arch.conf
10 f548183aeb29921c995b625a93c4acd3ef7faaec ima-ng sha1:feb140057713c4f1e383d79b71f6efdafbed7476 /conf/initramfs.conf
10 de528d81c1c203a597c313f54bbe45d54fd0cc18 ima-ng sha1:2231aa397f5b6327973d8fcaf540735fd1e39496 /conf/conf.d/resume
10 cf9a07066457e26219a6f345957a727b07096d8b ima-ng sha1:2199e965dcc97c6814b78528e5a5e690a29c0fd5 /scripts/functions
10 246635237cb7beaec50809203292f8623db6a83f ima-ng sha1:c7c7f8b3ae433ebe08189f143840f737d7711936 /scripts/init-top/ORDER
10 d0dc06f1a392d4505448572cd520b1ba6e53ff14 ima-ng sha1:4975101256fea3bf1e9a6a9ea5a4d97947f4097d /scripts/init-top/all_generic_ide
10 e2aab17444614530ec77595ef3f361bb00490100 ima-ng sha1:76dfee4b97d5327820a87ad4ec99a132a5f32cca /scripts/init-top/blacklist
10 a3dd75cea37a4330c6abefdeaa291feace1ee3a4 ima-ng sha1:869c43fa9e2c561d612c657ff45eb743beadc873 /scripts/init-top/ima_policy
10 465108cd35c590785a52eaecd9e997a0f570ada5 ima-ng sha1:a3f4886df912c0550f4e32cec1814e7f92e0218b /sbin/init
10 c78f4cecff4b004c9956c84628e6514a4d39881d ima-ng sha1:d11ce2e31ab441be705df3061a3d6fb7e41a504e /lib/x86_64-linux-gnu/ld-2.19.so
10 847203248af633d214e91dd1b3397e9d462771c7 ima-ng sha1:26837b475d0fb26d4256ce1744f52b264d67b58f /lib/x86_64-linux-gnu/libnih.so.1.0.0
10 367f76edbab585e2441bed7ee66fab6c7a1c0dad ima-ng sha1:d52c92a8019c259f40ae1240372dd598c2a1c54c /lib/x86_64-linux-gnu/libnih-dbus.so.1.0.0
10 b35e07f368b2d129dc9f3fd8ae325a9e3cf01a36 ima-ng sha1:d3892d8e70b27c4638ca8fbeceeed0386b7d672e /lib/x86_64-linux-gnu/libdbus-1.so.3.7.6
10 465a4a6342823c30427ca8374de54acb26bbb9fb ima-ng sha1:580764ad1cb67e7c37f49581ebf6369456795440 /lib/x86_64-linux-gnu/libselinux.so.1
10 5e8baf31a7f08a8e103f0f8174a3432e39161262 ima-ng sha1:91de58ef6be75cf952caecab0f2830c5b3527bbc /lib/x86_64-linux-gnu/libjson-c.so.2.0.0
10 d482b0fa3c1755c99380c279d73b77088c2a5d62 ima-ng sha1:011ea7ea14e6874e9da0245e4e6ed472d02814ed /lib/x86_64-linux-gnu/librt-2.19.so
10 a2733a6feac3a4d293af84f2ce47c1305cabc870 ima-ng sha1:65228a2bbff8ca52d2040ac55499b348f648cc81 /lib/x86_64-linux-gnu/libc-2.19.so
10 5da2378816b820601c8c708614784a7b5de5e8b8 ima-ng sha1:9ecd4089b74f1036c9825c2d082356e9ffb964f3 /lib/x86_64-linux-gnu/libpthread-2.19.so
10 cb8fc9859356d3802b365108d4a8baadf9251135 ima-ng sha1:9afccef2b8c4944cd78d25b87bc9198a3cb82406 /lib/x86_64-linux-gnu/libpcre.so.3.13.1
10 a3d30aa5bc7a24c3dd341d2eaa2ae4824915245a ima-ng sha1:cf26e327ee6f69694b080ae66c2572a6cb9c9c66 /lib/x86_64-linux-gnu/libdl-2.19.so
10 8b39d375a031075939a1621b2b470d0284c1f534 ima-ng sha1:c799f2ccebf69f87afc91520793631b3f0b9692b /lib/x86_64-linux-gnu/libnss_compat-2.19.so
10 ffab1636ff997c9b5040b637fe1cbfeae36988a5 ima-ng sha1:b74430744e6927384b34fd93385f8229b53e2dd7 /lib/x86_64-linux-gnu/libnsl-2.19.so
10 980f0b3422677f12d5af8850067e0b777358a013 ima-ng sha1:7fe4a578af95b0ebf1426573d088f110e5cdd8fe /lib/x86_64-linux-gnu/libnss_nis-2.19.so
10 60bd11e71fcd550996d557efaf1206832fe60cc5 ima-ng sha1:e12cc6838353f93bf43663081293d5891479f96f /lib/x86_64-linux-gnu/libnss_files-2.19.so
10 214c1d89e94ef8e89248a9b010cb7c050b6eef37 ima-ng sha1:8599d27418cf321a855d0c79091f1dfd5bec202d /bin/hostname
10 cb69d6e743aa7b96f011e7b74a37493bca7c5c26 ima-ng sha1:647437c3d7543c7c8d381903834c9ef42eb4cf69 /bin/sh
```

X18726-020317

Figure 7: **Integrity Measurement Architecture Evidence Log**

Trusted Platform Module

Documentation on TPM functionality is provided by the Trusted Computing Group (TCG), beginning with the TPM Main Specification [Ref 4]. TPM 1.2 was the most commonly used TPM in 2016. The Infineon OPTIGA SLB9670 TPM supports TPM 1.2 and 2.0. TPMs are very small, cost-efficient devices that provide root of trust for reporting (RTR) and root of trust for storage (RTS) security. This application note focuses on the RTR in which the measurement log file held in the TPM's PCRs is reported to the server.

In addition to support for RTR, TPMs provide capability that might be useful in Zynq-7000 SoC applications. TPMs provide re-programmable non-volatile memory. The TPM hardened cryptographic functions allow a key to be securely transmitted to the Zynq-7000 device on demand. TPMs provide a random number generator (RNG). RNGs can be used to generate keys.

The TPM RTR support operates within the IMA framework, providing significant security enhancements. When a TPM is added, the server's remote attestation of a client is based on a quote. A quote is measurement or evidence on the partitions booted. In TPM 1.2, an SHA-1 digest is used as the measurement for partitions loaded. In TPM 2.0, an SHA-2 digest is used as the measurement log for partitions loaded. The SHA digests are stored in the PCRs. Figure 8 shows the server-client communication for remote attestation.

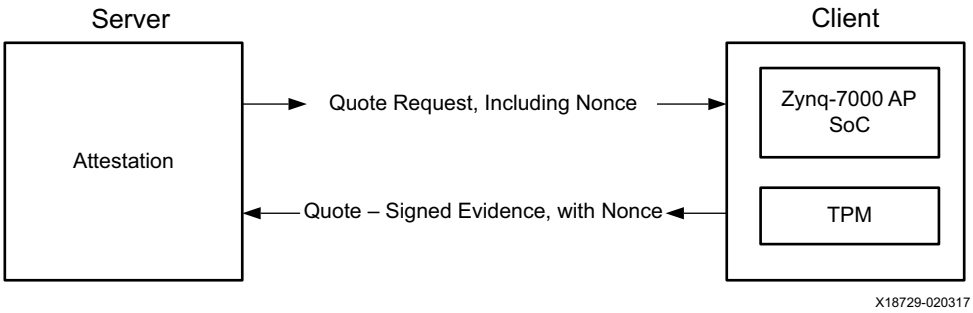


Figure 8: **Remote Attestation Using a TPM**

The flow in Figure 8 is outlined below.

1. The strongSwan attestation server requests a quote from the client. When requesting the quote, the server sends a nonce, which is a random number used to protect against playback attacks.
2. The client or Zynq-7000 SoC/TPM generates the evidence for the partitions loaded. The SHA-1 hashes are stored in the TPM PCRs. The SHA-1 of the BootROM code is stored in PCR[0], and the SHA-1 digest of the FSBL is stored in PCR[4].
3. The Zynq-7000 SoC/TPM client sends the quote to the server. This includes signed evidence and includes the original nonce.
4. The strongSwan server appraises the quote and, based on the results, follows a policy setup by the system administrator.

Zynq 7000 SoC-TPM Interface

The Zynq-7000 SoC-TPM interface provides the communication between the Zynq-7000 device and the Infineon OPTIGA SLB9670 TPM. The interface uses commands from a tpm_toolbox. The tpm_toolbox supports the following categories of commands:

- PCR reset
- Physical presence
- Get capability
- TPM startup/activate/physical enable
- PCR read/PCR extend

There are multiple commands in each category. A subset of the commands is used in the reference design. The Zynq-7000 AP SoC connects to the SLB9670 TPM using the SPI bus. The Zynq-7000 AP SoC contains a hardened SPI IP in the PS and a soft AXI SPI IP in the programmable logic (PL). The PS SPI is used in the reference design because it saves PL resources.

Figure 9 shows SPI-TPM functions implemented in the FSBL for the reference design.

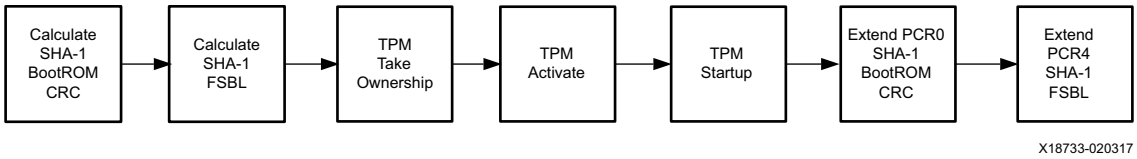


Figure 9: **FSBL TPM SPI Driver Functional Diagram**

In the measured boot reference design, the FSBL is modified to calculate the SHA-1 of the BootROM and the FSBL, and then extend the SHA-1 digests into the TPM's PCRs. The SHA-1 values are calculated in `sha1.c`. Code to take ownership and activate the TPM is in `slb9670_tpm_spi.c`. The PCRs are extended in `slb9670_spi_tpm.c`. Other files added to `fsbl/src` include `tpm_tools.h`, `tpm_tools.c`, `tpm_spi.c`, `tpm_spi_tis.c`, and `tpm.h`. Because BootROM code is not accessible by the FSBL, the SHA-1 calculated for the BootROM is calculated on the cyclic redundancy check (CRC) written by the BootROM code.

The FSBL TPM driver can be encrypted when stored in NVM and then decrypted and run from OCM. The reason for the FSBL extending the TPM PCRs with early load measurements is to limit the malicious attacker's time to change the code.

In the Avnet Starter IIoT board, the PS SPI interfaces to the SLB9670 Pmod using an MIO connection. To drive the pin reset of the TPM, the Zynq-7000 AP SoC hardware design includes a PS GPIO which is used to drive the TPM reset pin. The ResetTPM function is in `main.c`.

Network Security in Measured Boot

Software updates and remote attestation require a secure connection between a server and the embedded system clients. The network has a large attack surface because it can be attacked by any adversary with access to the Internet. For firmware updates, a server to client(s) connection is used. In some factory automation environments, client-to-client communication is also needed to coordinate operational procedures.

Figure 10 shows a high-level view of strongSwan's implementation of TCG's trusted network connect (TNC). In the TNC architecture, the server connects to clients in the integrity evaluation layer. The client does integrity measurement collection (IMC), and the server does integrity measurement verification (IMV). The software on the client is the platform trusted service (PTS), trust software stack (Trousers), and the TPM tools.

PTS uses the Trousers library to access the TPM and IMA measurements. The reports use standard PTS formats for interoperability between applications and vendors.

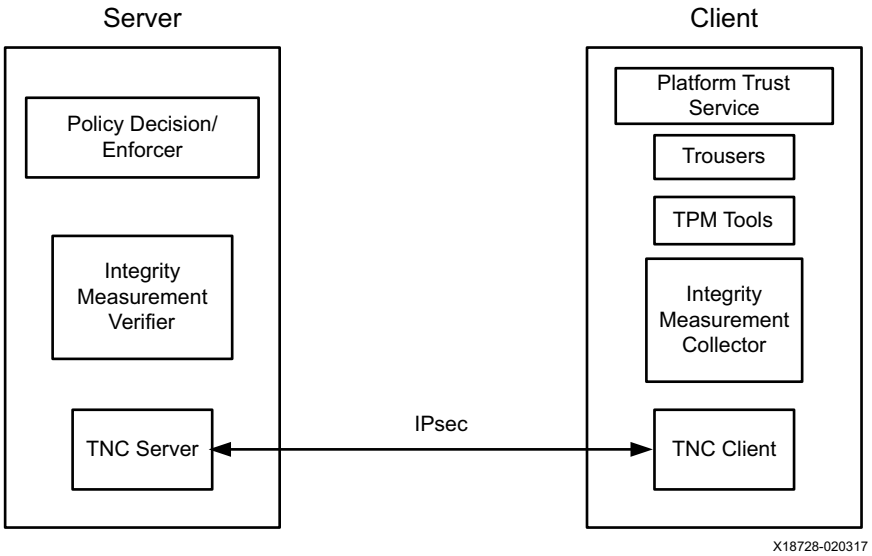


Figure 10: **Trusted Network Connect for Remote Attestation**

The policy decision point (PDP) defines the action taken by the server after measurement verification. A typical policy/action is to limit network access until remediation is done. A different policy is used when availability is a critical driver, and some out-of-range measurement verifications are not treated as critical. The TPM Main Specification [Ref 4] provides an overview of the TNC architecture.

The underlying security for TNC in the reference design uses IPsec. This includes conventional technology such as Internet key exchange (IKEv2), public key infrastructure (PKI), and the transport layer security (TLS) handshake in which the encryption and authentication algorithms are negotiated and pre-shared keys are exchanged. A virtual private network is set up in the strongSwan architecture. A privacy CA generates the x509 certificates. The strongSwan `Readme.txt` provides information on the IPsec flow.

Reference Design Functional Overview

The following steps are done in the reference design to set up IMA, TPM, and network security.

- Activate IMA in the Linux kernel
- Configure the IMA policy
- Activate the TPM
- Set up the privacy certificate authority (CA)
- Set up the attestation client (Zynq-7000 AP SoC)
- Generate an attestation identity key (AIK)
- Configure the integrity measurement collectors
- Configure the TNC client
- Configure the VPN connection
- Set up/configure the attestation server (strongSwan VPN/TNC server)
- Collect measurement values
- Register the device with the policy manager

The process is defined on the strongSwan website.

Conclusion

Zynq-7000 AP SoCs provide significant advantages in their ability to program both hardware and software on the same device. Cost-effective firmware updates are a key to increasing embedded system capability and providing maintenance to reduce the TCO. Remote firmware updates rely on using the Internet, opening the embedded system to cryptographic attacks. This application provides mechanisms that provide proven-in-use security for connected devices.

References

1. *Zynq-7000 All Programmable SoC Technical Reference Manual* (UG585)
2. *Run Time Integrity and Authentication Check of Zynq-7000 AP SoC System Memory* (XAPP1225)
3. Linux Integrity Measurement Architecture wiki.strongswan.org/projects/strongswan/wiki/IMA
4. TPM Main Specification www.trustedcomputinggroup.org/tpm-main-specification

5. *Secure Boot of the Zynq-7000 All Programmable SoC* (XAPP1175)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/07/2017	1.0	Initial Xilinx release.

Please Read: Important Legal Notices