APPLICATION NOTE

DISTRIBUTED SECURITY WITH COTS CPU BOARDS

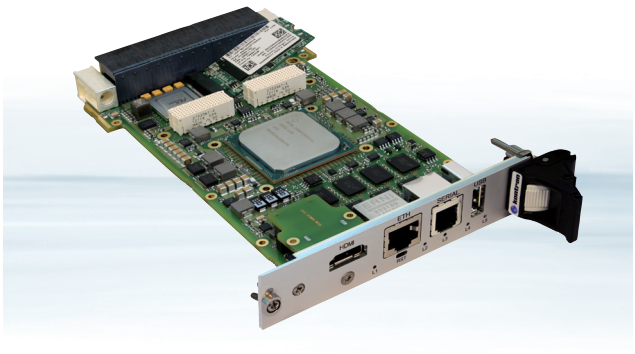
POSSIBILITIES START HERE

kontron
S&T Group

CONTENTS

Highly sensitive Embedded Applications can benefit from secure elements available on COTS CPU boards to build a robust digital security solution. By distributing a secret in different circuits, the risk of a successful attack is significantly lower, the discovery of a vulnerability is less impacting and the security rules associated with the provisioning of the keys is simplified.

Author:
Serge Tissot, Technical Strategy Manager
at Kontron Modular Computers, France
support.KFR@kontron.com



// The figure below shows the Kontron VX3058 3U VPX module featuring a Intel Xeon®-D1500 processor, up to 8 CPU cores, DDR4 main memory and the two mentioned COTS secure elements.

Hardware enforced protections bring a significant contribution to the security of computer systems whenever it is assumed that a hacker could take control of a system one way or another, including through operating system vulnerabilities or password disclosure. In such case of an attack, you still want to make sure that secrets governing the authentication of the machine, the confidentiality and the integrity of critical data, which is stored locally or being sent over a network, are still preserved.

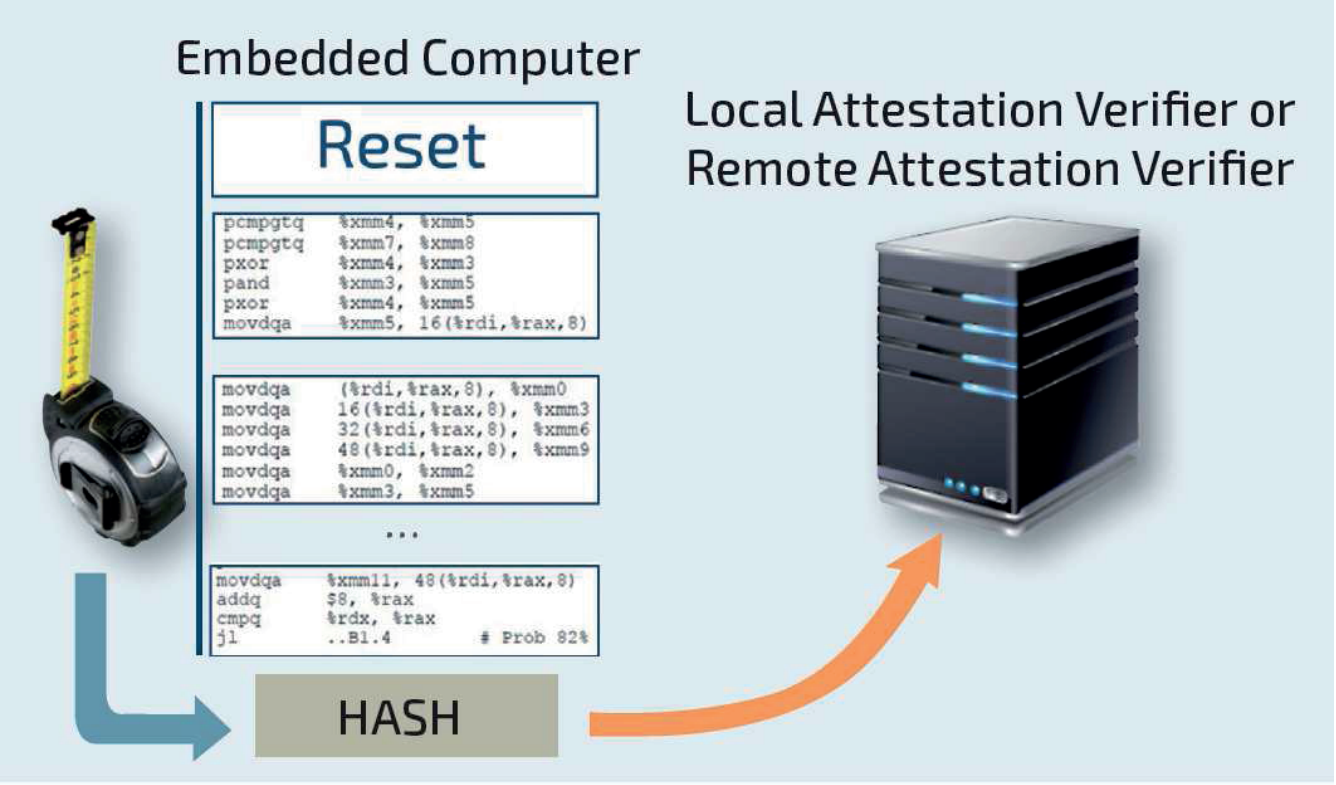
It is typically the main role of secure elements to preserve the most important secrets. Not only those dedicated hardware circuits are storing the secret keys, but they are also able to use those secrets internally to perform the expected operations like encryption, decryption, signature, key generation, without having to expose the secret keys externally at any time. Moreover, the secure element silicon dies are designed to be resistant to hardware side channel attacks like electromagnetics probes, current consumption probes, laser beam, out of range operating temperature, etc.

Modern COTS CPU boards are equipped with such hardware secure elements. Let us take the example of the Kontron VX3058 3U VPX Intel Xeon® CPU module featuring two secure elements:

- The TPM (Trusted Platform Module) secure element is used for the security of the boot of the system and the authentication of the computer over the network during TLS (Transaction Layer Security) connections
- The Kontron APPROTECT security solution based on Wibu-Systems CodeMeter® secure element is dedicated to the integrity and the confidentiality of the application.

Trusted boot to detect system software alteration
Detecting any unexpected change of code, data or configuration during the boot process, starting from reset time, is a highly desirable feature to discover attempts to compromise the system.

The trusted boot mechanism relies on the TCG (Trusted Computing Group) international standard, where a Trusted Platform Module (TPM) secure element is used to verify the validity of the boot of a computer. The TPM measures with a hash value all pieces of code executed during the boot, including firmware, firmware setup, bootloader, operating system and more if needed. Any single bit changed in the boot, as compared to a reference code, can be detected and reported in a local or remote cryptographic attestation. This scheme is illustrated below.



In case of a discrepancy detected, it is up to the application solution to decide how to block further activities or to accept a new hash reference in the case of a fresh update of the boot code.

Authentication with TPM to secure network protocols
To establish a secure network connection like https, a private key is needed on the server side at least, and also recommended on the client side for embedded computing since no operator is present to enter a password.

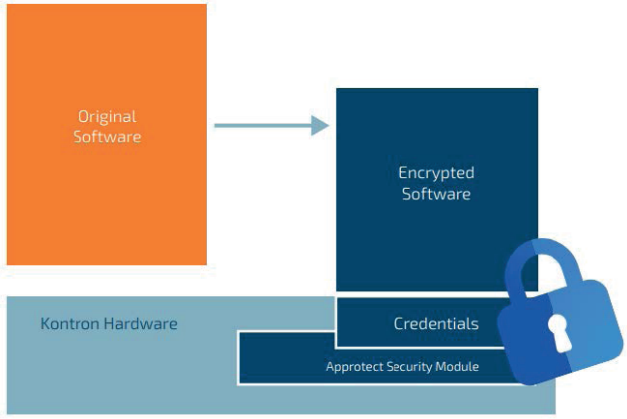
The TLS connection starts with an authentication phase to make sure that the distant server (and client in case of strong authentication) is the expected one, as advertised in the URL bar of the browser for example. For this step, an asymmetric cryptography algorithm like RSA, using a public and a private key, is used. The server authenticates itself and proves to the client that it owns the (secret) private key corresponding to the public key of this URL by signing with its private key. Then the client can, in confidence, encrypt with the server public key, session keys to continue exchanging with a lighter symmetric cryptography protocol like AES. In this process, note that a certificate is provided by the server to establish the correspondence between the public key and the URL; this certificate is signed by a root authority in order to control its validity (this is the PKI public key infrastructure process).

It is often NOT a good idea to store the private key on the disk, even encrypted, because at some point, it will be decrypted in memory and CPU registers when used at the beginning of a network connection. With the TPM authentication, the private key will be stored and used under the hardware protection of the TPM, so that it is never exposed. Without this hardware protection, the risk exists of having the private key stolen, which would allow the duplication of a compromised server or client machine, or a man in the middle strategy spying the communication.

Kontron APPROTECT security solution (Wibu-Systems CodeMeter®) for protection of the application

The main security threats at the application level are:

- **Integrity**: the running application might be hacked or patched in binary, on the disk or in memory, to modify its behaviour or work around some checks.
- **Confidentiality**: the way the application code is working could be analysed in details by looking at the execution code, in order to learn or to reproduce its behaviour
- **Unauthorized copies**: the embedded systems (software application or full equipment) suffer the risk of being cloned without authorization.



The Kontron APPROTECT security solution (Wibu-Systems CodeMeter®) protects the application from those risks with the security of a dedicated hardware secure element located on all Kontron boards. All or part of the application code is encrypted on the storage device and will only be decrypted in memory in presence of the keys stored inside the secure element. The encryption of the code can be done by Kontron or by the customer.

At run time, the integrity of the application in memory is permanently checked. The application can be fully encrypted as a whole with no modification, or only critical part can be encrypted and on the fly decrypted by using a dedicated API, providing the additional guaranty that a whole decrypted code image will never be available in memory at any time. It is possible to restrict the execution of the protected application to a particular board serial number identified by its APPROTECT secure element, to avoid unauthorized software copies.

Besides of those application protection dedicated functions, an API to access general cryptographic primitives of the secure element is available, including symmetric and asymmetric cryptographic functions.

SEC-Line security product line from Kontron

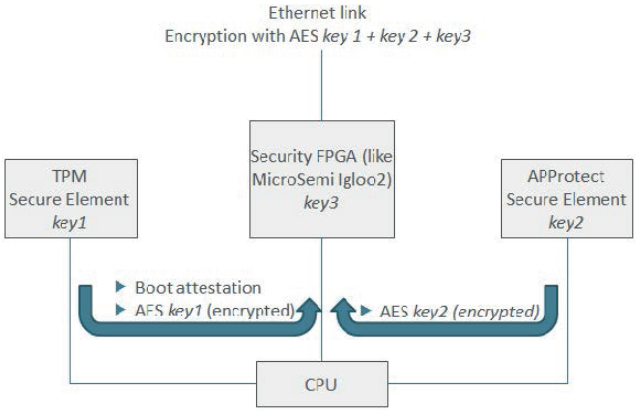
The three hardware enforced security mechanisms describe in the sections above are part of the SEC-Line security product line from Kontron, running on Kontron boards: trusted boot, authentication with TPM and Kontron APPROTECT security solution (Wibu-Systems CodeMeter®). Please refer to the SEC-Line datasheet for more details.

This product line also includes a software only mechanism, secure boot, by which the Bios firmware restricts the boot capability to signed images having a certificate registered from a Bios menu. As opposed to the trusted boot, the secure boot will block the boot of incorrect images, whereas the trusted boot will let the boot happen and later decide to stop at the application level in case of compromised boot code.

DISTRIBUTED SECURITY FOR HIGHLY CRITICAL APPLICATIONS

For highly sensitive applications, it is often desirable to combine redundant and independent security means to achieve the level of security required, so that if a single mechanism gets compromised, the system integrity is preserved.

This application note, as an example, describes how secure elements of standard COTS CPU boards, such as those listed in the previous chapter, can be combined into a dedicated security FPGA to crypt a critical Ethernet link of the machine. The main principle is shown on the diagram below:



On this diagram, the final symmetric encryption key (for example an AES256 key) protecting the Ethernet link can be a XOR combination of Key1, Key2 and Key3. Key1 would

originate from the TPM secure element and transferred in an encrypted form to the FPGA by the CPU. Similarly, key2 would originate from the APPROTECT secure element and transferred to the FPGA by the CPU. Key3 would always stay resident inside the security FPGA. Finally, the boot attestation issued from the trusted boot mechanism, to verify the integrity of the boot of the firmware and operating system, would be sent to the FPGA for checking the boot sequence.

A more detailed description of the low level mechanisms involved for the TPM, the APPROTECT secure element and the FPGA is provided in the next chapters.

TPM SECURE ELEMENT CONTRIBUTION

Trusted boot aspect

A general view of the trusted boot mechanism is described in chapter 2. In this section, we provide additional precisions on how it operates and how it can be used with an FPGA.

The TPM features a moderate speed slave serial interface like LPC bus or SPI. The hash of the different code & data segments executed will be calculated internally by the TPM and stored internally to the TPM as PCR registers. It should be noted that the TPM1.2 is limited to SHA-1 128 bits hash algorithm whereas the TPM2.0 can use the SHA256 more robust algorithm. A first set of PCR registers is used for measuring the Bios, a second set is used for measuring the boot loader, the operating system and more if needed.

It is the software itself that submit to the TPM the binary code and data segments to be measured, like a DMA. But to avoid any chance to submit misleading information to the TPM, those code segments are always measured before execution, and this is even true for the initial segment containing the very first Bios instruction after reset, thanks to a CPU internal microcode.

At the end of the process, the TPM will sign internally the PCR values as a whole, together with a random value provided by the requester of the attestation, in order to prevent the re-use of an old attestation. The CPU software will then build an attestation file, for example in XML format, which can be exposed without any risk, so that the verifier party can compare the hash values to a reference attestation established initially. The signature by the TPM and its verification by the FPGA will typically be performed using RSA 2048 key pairs.

In our case, the verifier party is the FPGA itself. The FPGA provides, as a slave interface, the random value to the TPM (through the CPU), the TPM internally signs the hash and random values, and finally, the CPU transfers from the TPM to the FPGA the signed attestation made of measured boot hash values and random value. The FPGA verifies that the random value is the one associated with its request (challenge mechanism) and can control the correctness of the measurement of the boot by comparing with its internal registers holding the correct provisioned hashes. The figure below shows an example of an XML attestation file.

```
- <Report xmlns:core="http://www.trustedcomputinggroup.org/XML/SCHEMA/1_0_1/core_integrity#"
    xmlns:stuff="http://www.trustedcomputinggroup.org/XML/SCHEMA/1_0/simple_object#" xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.trustedcomputinggroup.org/XML/SCHEMA/1_0/integrity_report#" ID="IR_dbc1a96e-b8cb-11e7-8e5c-00b3386ee198" UUID="dbc1a96e-b8cb-11e7-8e5c-
    00b3386ee198">
  - <QuoteData ID="TBD">
    - <Quote2>
      - <QuoteInfo2 Tag="S4" Fixed="QUT2" ExternalData="urZ6jPrL7kYHHX4gjh/bzM8+Rxc=">
        - <PcrInfoShort>
            <PcrSelection SizeOfSelect="3" PcrSelect="/ScA" />
            <LocalityAtRelease>1</LocalityAtRelease>
            <CompositeHash>TFMaNfvBhow5Z6DVTM+gpCWg2Qw=</CompositeHash>
          - <PcrComposit>
              <PcrSelection SizeOfSelect="3" PcrSelect="/ScA" />
              <ValueSize>0</ValueSize>
              <PcrValue PcrNumber="0">I2pFt7Xc3j1sG7qFnndyPTJEsOk=</PcrValue>
              <PcrValue PcrNumber="2">BP3hO8jczbTkcdKxSJfuHkUcRQo=</PcrValue>
              <PcrValue PcrNumber="3">Oj94DxGktJlp/KqAzW45V8M7InU=</PcrValue>
              <PcrValue PcrNumber="4">jr9/PQ0M8rKzEYipzq6mawoMXYU=</PcrValue>
              <PcrValue PcrNumber="5">qb0dpHLa1qOt7E53ObEI5401uag=</PcrValue>
              <PcrValue PcrNumber="6">eM13WYZqd9AxA8IDW/fcfmHcGS4=</PcrValue>
              <PcrValue PcrNumber="7">Oj94DxGktJlp/KqAzW45V8M7InU=</PcrValue>
              <PcrValue PcrNumber="8">0/bJhRQn1An0d/n0mN3DWzx6hOQ=</PcrValue>
              <PcrValue PcrNumber="9">TCa5i5AOrYMqjjSrCpz+7+CncNk=</PcrValue>
              <PcrValue PcrNumber="10">56iCOb4K0HIvaKyfxsQoJj/qGX0=</PcrValue>
              <PcrValue PcrNumber="13">2H8g320uc10j11SOM/jsgcb1mMQ=</PcrValue>
            </PcrComposit>
          </PcrInfoShort>
        </QuoteInfo2>
      </Quote2>
    - <TpmSignature>
        <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
        <SignatureValue>gIQB/rdLLSzJjT5McmEhjPmtGQraRbhmuFQ1s5s2MAN3kendIWWbiwI8zX+IVCmvQKkQq9bh
          rAocm5NXoKeTfuqBVY74jUit7eozot9BLcVTtbJXcI+XYHlg/pSQryFaMr7mbXPwc3Yhau4r
          4UcYnpvl5iGHxvqdcwJDK+uHRxCPp6mpdjHCSYGeID7FhaxZix4DWck2K5/4Dnfj5rhtuSwZ
          1WHoVF4zNgNHWH5nW0uG3po16TMOOqnjevOOuq6gtiQtuXU/HBC3JnuUeCddo1WPODi/JV2H
          q9FDWXe9q7u88xLsC8jj+bOOlAKrnNv4cEGjC8j9+f5xVPOsI5DoyA==</SignatureValue>
```

Key1 transfer from TPM to FPGA

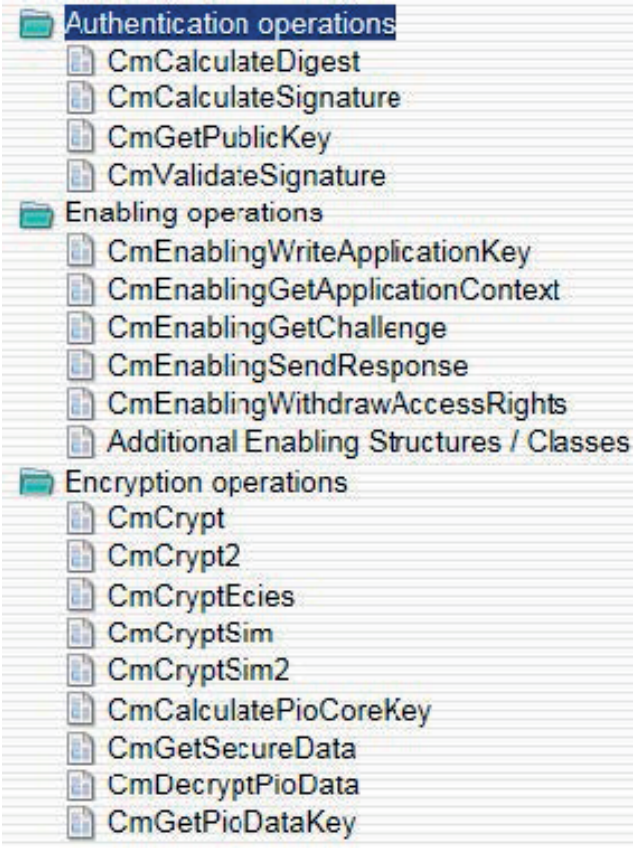
In our example, Key1 is the partial AES symmetrical key that will be used by the FPGA to reconstruct the final AES key protecting the Ethernet link. This key is provisioned into the TPM, or into a so called data store on the disk protected by the TPM, and must be passed securely to the FPGA.

To implement this function, key1 is encrypted at provisioning time with a public key, using typically an RSA 2048 algorithm. Similarly to the case of the trusted boot, the RSA encrypted key1, together with a random number provided by the FPGA, are authenticated by a TPM signature. This information is passed to the FPGA through the CPU.

The FPGA will then check the TPM signature and the correctness of the random number it initially provided. After this step, the FPGA is the only one that can decrypt Key1 with its private key, in order to retrieve the Key1 value.

APPROTECT SECURE ELEMENT CONTRIBUTION

The APPROTECT secure element, as described in the first chapter, contains a firmware and have a backend software dedicated to the security of the application. It provides a core API able to perform cryptographic functions, including symmetric encryption like AES and asymmetric encryption based on elliptic curve cryptography, like ECDSA, for signing. The figure below illustrates the API.

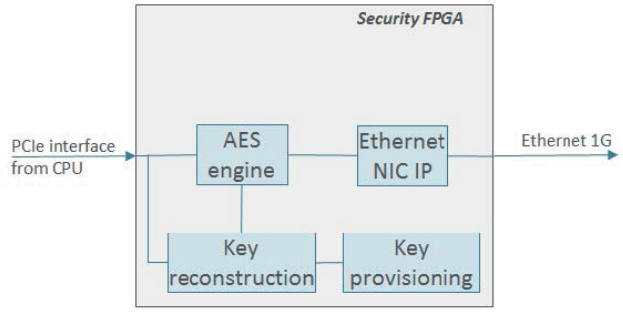


By using those primitives, a similar process that has been described in the previous chapter for transferring Key1 from the TPM to the security FPGA, can be used to transfer Key2 provisioned in the APPROTECT secure element, to the security FPGA.

The Key2 encrypted and a random number generated by the FPGA are signed by the APPROTECT secure element with the ECDSA algorithm. The CPU transfers the resulting data and signature to the FPGA which will verify the signature and decrypt Key2 for its further internal usage.

SECURITY FPGA CONTRIBUTION

In our example, a dedicated security FPGA is handling the network interface, using the PCIe as the link to the CPU, and an internal Ethernet IP to connect to the network. Using an FPGA family such as MicroSemi Igloo2 is a good choice for the FPGA since it implements cryptography hardware IPs and has some built in security attributes such as on chip flash memory to program FPGA cells and an authentication mechanism to accept a new FPGA code. The diagram below illustrates the internal blocks of the FPGA.



The dedicated security FPGA can for instance, equip a VX3058 XEON 3U VPX module from Kontron, by using the top side M.2 card featuring the required PCIe host interface and the Ethernet I/O connectivity (1000Base KX) routed to the rear of the module.

CONCLUSION

We have described in this application note how standard hardware security mechanisms on COTS CPU boards can be leveraged to build a distributed security scheme with critical keys stored in distinct secure elements. This allows the support of highly critical applications where a single point of weakness, eventually unveiled in the future, would not be enough to compromise the system.

The use of a dedicated security FPGA to sequence and gather the keys have many benefits, like the possibility to use non standardized crypto algorithms, the ability to customize different algorithms for different customers and the guaranty to have better control on the risks related to the developers themselves.

About Kontron – Member of the S&T Group

Kontron is a global leader in IoT/Embedded Computing Technology (ECT). As a part of technology group S&T, Kontron, together with its sister company S&T Technologies, offers a combined portfolio of secure hardware, middleware and services for Internet of Things (IoT) and Industry 4.0 applications. With its standard products and tailor-made solutions based on highly reliable state-of-the-art embedded technologies, Kontron provides secure and innovative applications for a variety of industries. As a result, customers benefit from accelerated time-to-market, reduced total cost of ownership, product longevity and the best fully integrated applications overall.

For more information, please visit: **www.kontron.com**

GLOBAL HEADQUARTERS

KONTRON S&T AG

Lise-Meitner-Str. 3-5
86156 Augsburg, Germany
Tel.: + 49 821 4086-0
Fax: + 49 821 4086-111
info@kontron.com

www.kontron.com