WHITE PAPER

Bruno Johnson
CASCODA
b.johnson@cascoda.com

OPEN CONNECTIVITY FOUNDATION®

Secure embedded IoT using OCF and Thread

December 15, 2021

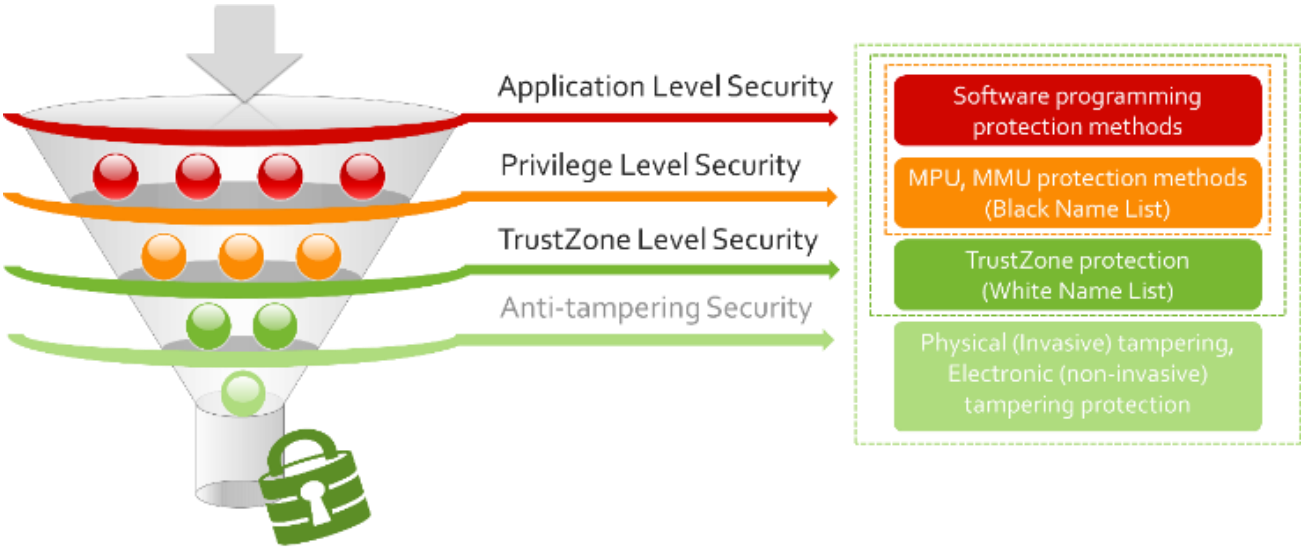
OPEN CONNECTIVITY FOUNDATION MEMBER CONFIDENTIAL

Introduction

To date, it has not been possible to perform secure IP-based IoT communications on a small low-power embedded module.

This is now possible is due to new secure connectivity standards combined with the development of secure low-power microcontrollers with cryptographic accelerators.

Security requires a multi-faceted approach, encompassing both hardware and software, as shown in Figure 1 below.



This Figure is reproduced from work created and shared by the Nuvoton Corporation

Figure 1 Security levels related to embedded products

The Open Connectivity Foundation (OCF) application layer running over the Thread low-power wireless network layer is an ideal combination to achieve this goal.

Moreover, this combination is very well suited to commercial deployments. For example, in a multi-tenant environment, Thread can be used by IT (Information Technology) administrators to create a self-forming wireless mesh network that spans the entire building, using network-level encryption to protect the data from outside observers. In the meantime, OCF can be used by the OT (Operational Technology) administrators to facilitate certificate-based access, with access control lists, to give secure access to the required individuals and administrators at the time that they may need it.

This combination can drastically reduce installation costs. The Thread mesh-network layer means that installation can be performed by non-specialist installers, while the OCF application layer can be configured remotely. Figure 2 below shows how such an installation may be configured.

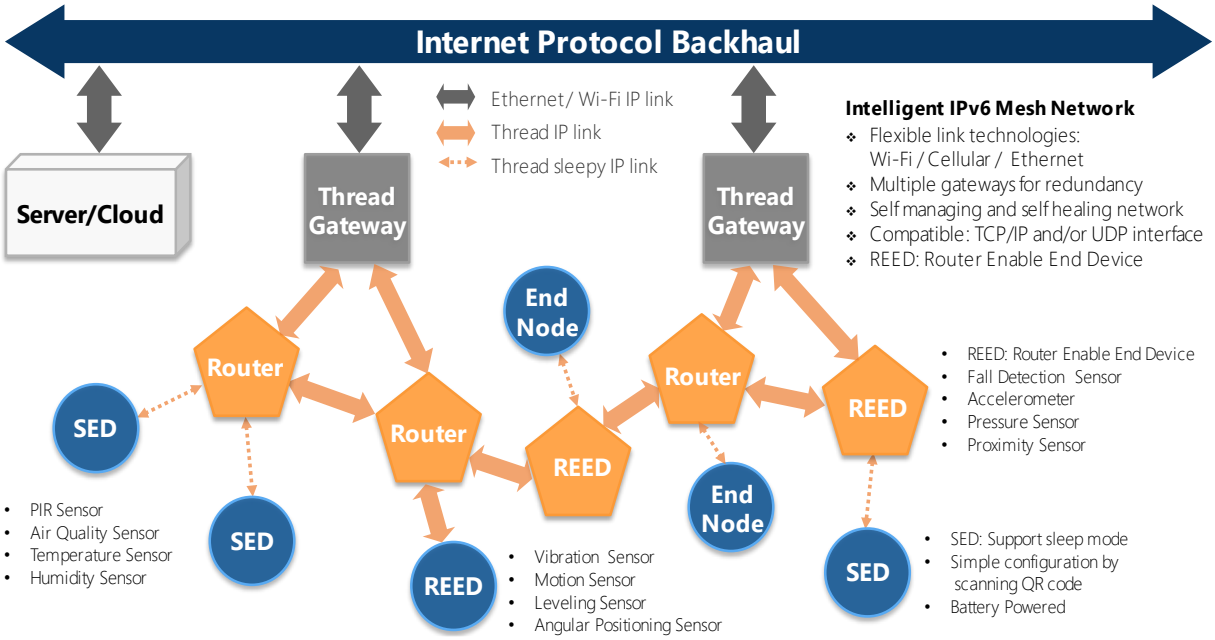
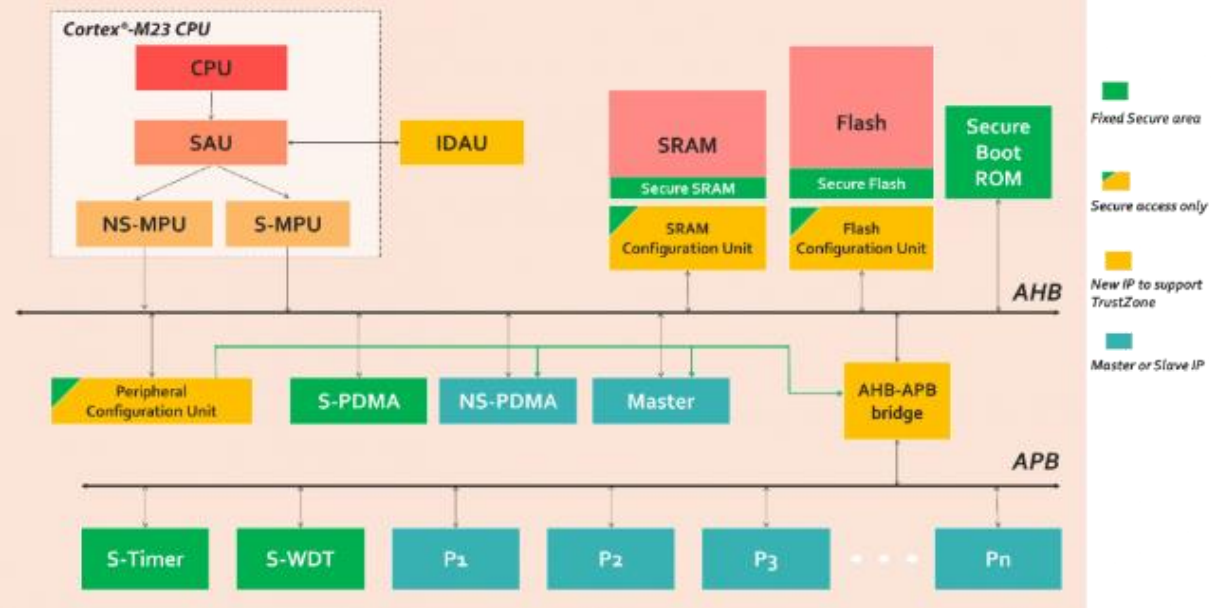


Figure 2 Chili2 OCF-over-Thread solution

In this white paper, Cascoda presents its OCF-over-Thread solution running on its Chili2 embedded module, as well as some of the challenges that were overcome in its development.

Technical Overview

The Chili2 module family is a fully-featured Thread-based wireless solution, using an ARM Cortex-M23 microcontroller, the M2351. The microcontroller features 512kB of application flash and 96kB of on-chip SRAM, and runs using a clock frequency of 48 MHz.



This Figure is reproduced from work created and shared by the Nuvoton Corporation

Figure 3 The M2351 microcontroller

Having only 96kB of RAM is an exceedingly strict constraint for the OCF application layer stack. The Raspberry Pi 3 included inside the OCF's Developer Kit has 1 GB of RAM, four orders of magnitude more than the Chili2 module has to work with. However, with a lot of effort dedicated to memory optimizations, Cascoda was able to overcome this limitation.

Thread is a low-power wireless IPv6 mesh network-layer protocol. The core benefit of using Thread as opposed to Wi-Fi is that it enables low-cost, battery-powered devices.

Additionally, the self-forming, self-healing mesh network allows for robust deployment of networks containing hundreds of devices.

Since Thread is application-layer agnostic, the Chili2 is able to run multiple application layers over Thread simultaneously, in this case a Domain Name System (DNS) and a Simple Network Time Protocol (SNTP), alongside OCF. The OCF application framework consists of the following layer-stack, as shown in Figure 4:

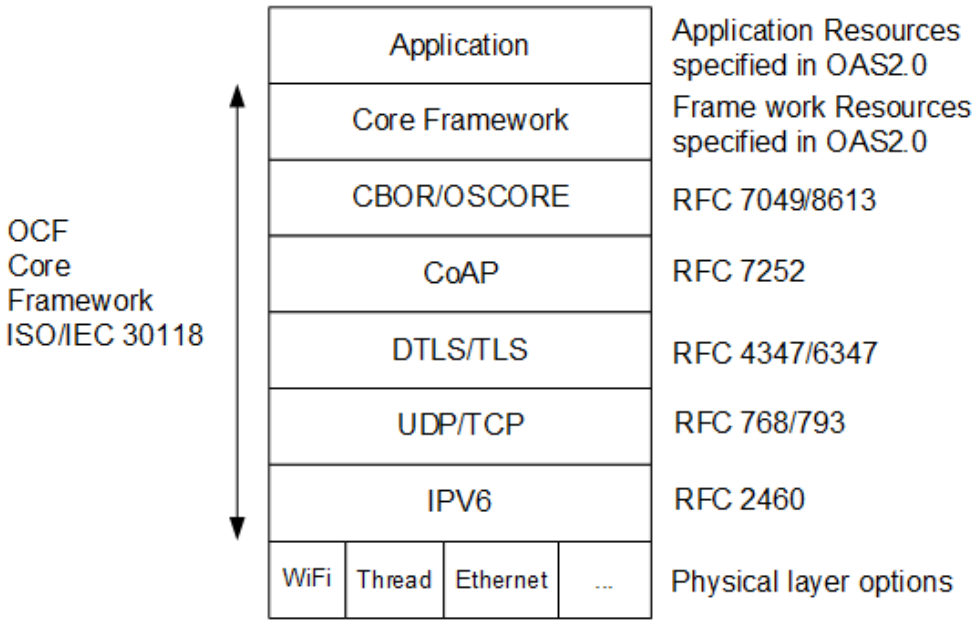


Figure 4 OCF application framework

OCF uses the Constrained Application Protocol (CoAP), an Internet Application Protocol for constrained devices. CoAP enables constrained devices to communicate with the wider internet by means of a REpresentational State Transfer (REST) interface common to most internet-connected applications.

This communication is secured with Datagram Transport Layer Security (DTLS), an internet communications protocol providing security for application-level messages. DTLS facilitates Public Key Infrastructure (PKI) for authentication, which uses certificates rather than an ID and password. After a DTLS connection is established, all further communication is both authenticated and encrypted using the state-of-the-art Advanced Encryption Standard (AES).

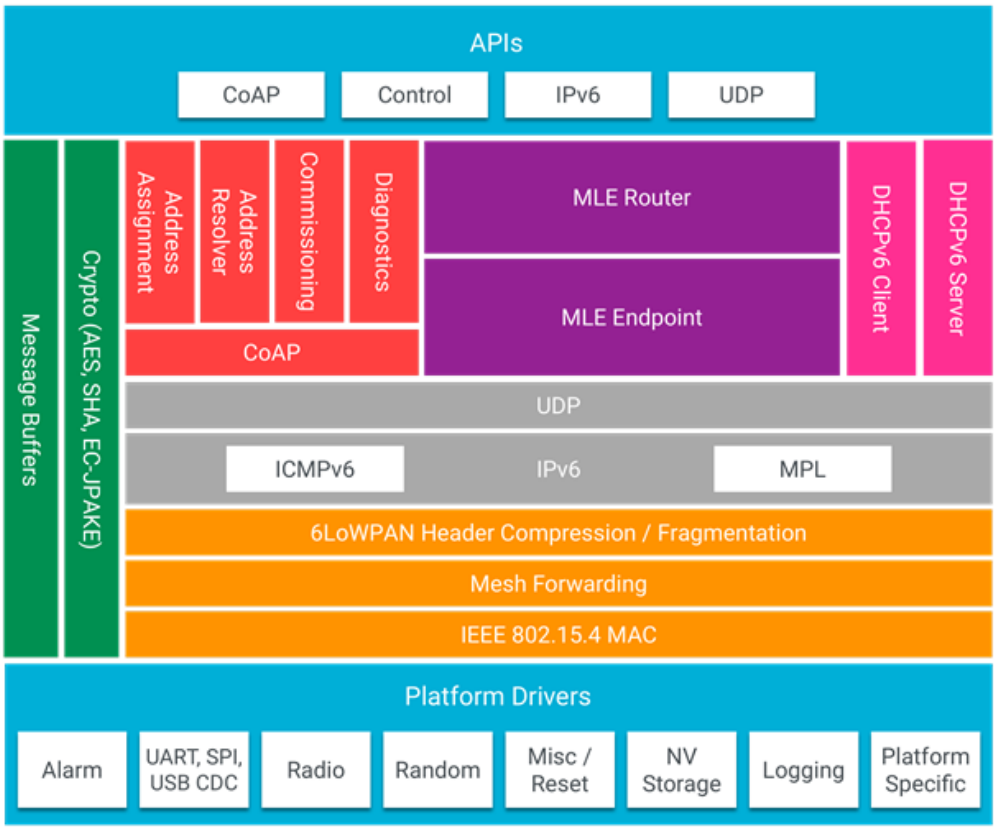


Figure 5 Thread component overview

The Thread network is a meshed network; it can contain border routers, which are akin to the network component that acts as a Wi-Fi access point, but for Thread-based devices. A border router need not be a single point of failure, as a Thread network can have multiple border routers. In addition, Thread also allows for multiple routers to create a meshed network which is self-configuring and self-healing, e.g., if a router node is removed from the network, the network will reconfigure itself so that all devices can still communicate with each other.

Aside from Thread border routers and routers, Thread facilitates two types of end device, namely the End Device (ED), and a Router Enabled End Device (REED). A REED can act as a router, while an ED cannot. REED nodes dynamically switch between the end-node and router functions depending on the requirements of the meshed network. This automatic switching means that Thread networks can be configured for a high degree of redundancy. Chili2

modules running OCF-over-Thread can be configured as any Thread node role; however, for the purposes of this white paper, the Thread network software is configured as a REED.

Since both Thread and OCF are IPv6 and User Datagram Protocol (UDP) based, the usage of OCF as an application protocol on top of Thread is a perfect match. The obvious overlap between OCF and Thread is a CoAP library. After investigation we did not merge the CoAP functionality. The CoAP stack in OCF and Thread are heavily integrated and tailored. For example, OCF CoAP method call-backs are integrated with Concise Binary Object Reapresentation (CBOR) encoding/decoding, which is not needed at the Thread level.

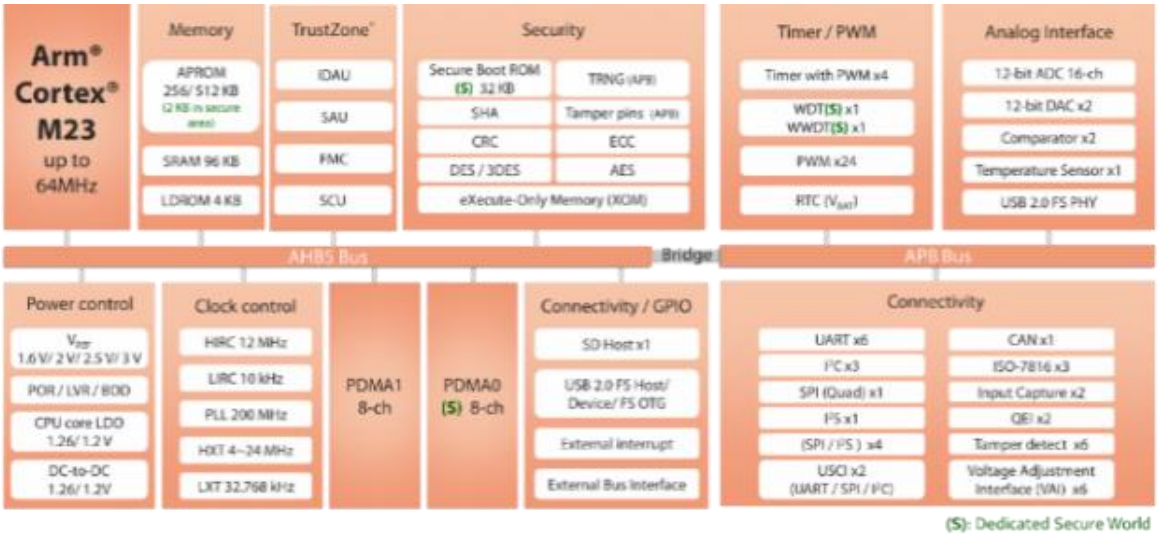
Furthermore, there was no need to do this optimization; there is enough program storage on the device to run both stacks. Note that the advantage of not doing this integration is that it will be easier to transition to newer versions of the open source Thread stack (OpenThread), and the OCF stack (IoTivity-lite).

Why the M2351?

At a glance, the Chili2 platform may seem underpowered, especially when compared to the Raspberry Pi.

However, it is a fraction of the cost and uses two orders of magnitude less power, allowing operation from coin cells and energy-harvesting transducers.

The M2351 features hardware cryptographic accelerators which are vital for the process of OCF application-layer PKI onboarding, i.e. securely connecting a new device, first to the network, then subsequently to its enable services. Without hardware acceleration, such a high level of security would be impractical to use on an embedded microcontroller. In addition, Cascoda used cryptographic acceleration to enable Thread commissioning and network-level encryption.



This Figure is reproduced from work created and shared by the Nuvoton Corporation

Figure 6 M2351 Block diagram

Cascoda uses the M2351's True Random Number Generator as the source of entropy for all encrypted communications, and for generating the secure storage key. Finally, the on-chip Real-Time Clock allows Cascoda to verify the validity of PKI certificates even in the face of network outages, when network time may temporarily be unavailable.

Next to all security features, the processor has several connectivity possibilities to connect sensors and actuators.

Memory partition

In order to gain an understanding of how memory is used by the software stack, we must examine the following memory regions:

- Static memory, which stores variables that persist across the lifetime of the program
- Stack memory, for storing local variables within functions
- Heap memory, for storing large variables that are used across several functions
- Code memory, for storing constant data and code that does not change during execution.

The execution code is stored in flash memory, and the M2351 executes the code directly from flash. This means that although optimizing the execution code is important, it does not affect the SRAM usage. The program's working data, including static memory, stack memory and heap memory, are stored in SRAM.

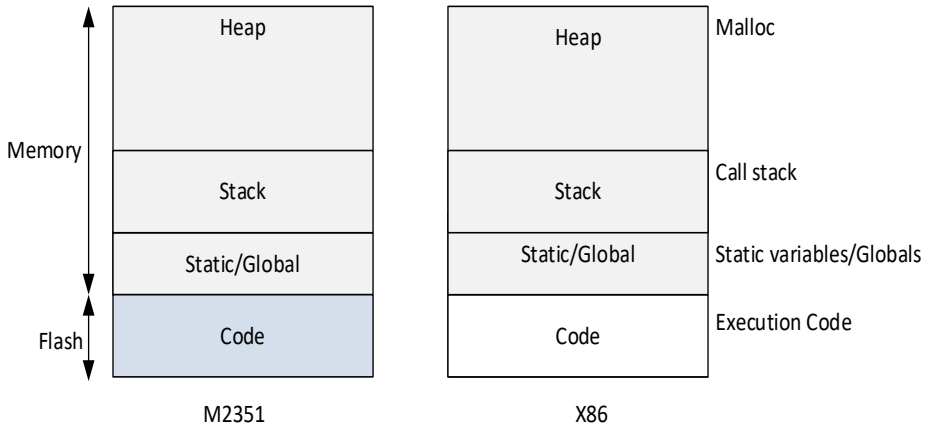


Figure 7 Memory partition

OCF uses both static and heap memory, while Thread uses static memory almost exclusively, and some static memory is also used by system functions. MbedTLS (an implementation of the Transport Layer Security (TLS)) generally allocates memory on the heap, alongside OCF. The stack memory region is shared by all components of the system. Since there is no clear separation between the three, stack memory is considered to be entirely shared. Finally, the application being profiled is a 'oic.d.test.module.actuator' device containing the three

resources needed for module certification. With these things in mind, Figure 8 below depicts the SRAM memory usage:

Memory User	Memory used (bytes)
OCF + mbedTLS (heap)	47104
OCF (static)	13698
Thread (static)	25846
System (static)	4578
Shared (stack)	6860
Application(static)	218

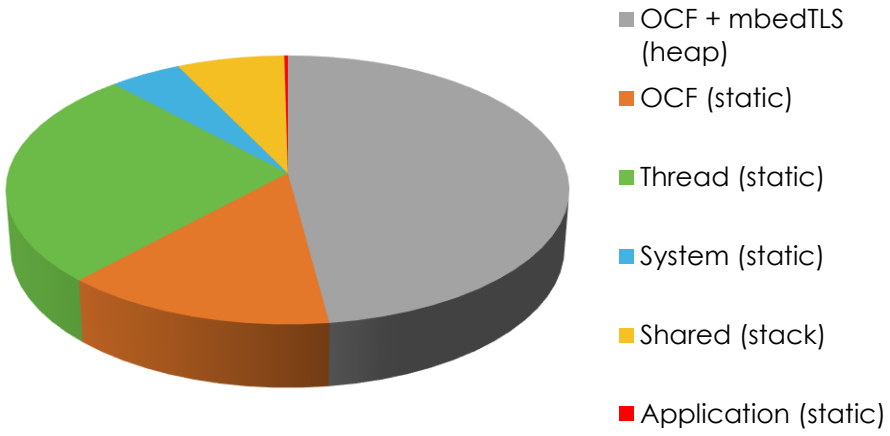


Figure 8 SRAM usage

OCF (together with mbedTLS) uses the most memory within this system at 46kB, or 48% of the memory available on the Chili2. Because OCF controls most of the allocations created by mbedTLS, its memory usage figures have been combined, as it is difficult to separate the two without extensive runtime profiling.

In the configuration profiled above, Thread is configured as a REED Thread device – a device that can route Thread messages. With this configuration, Thread statically uses 26% of the available memory. Thread also makes use of mbedTLS, but under Thread control, mbedTLS mostly allocates statically or on the stack.

An additional SRAM saving of 10kB could be made by configuring the device as a Thread ED instead of a REED. The drawback of this approach is that EDs cannot function as routers within the Thread network, which in some cases could limit some of the redundancy features of the Thread network. However, configuring the stack as a Thread ED frees up memory which could be used to facilitate complex applications within a mixed deployment where other devices can route messages.

Flash partition

Analyzing the use of flash memory is simpler – it can be done by inspecting the symbols within the compiled binary.

Flash User	Flash used (bytes)
mbedTLS	183814
Thread	121177
OCF	58553
System	61291
Application	4632
Free space	94821

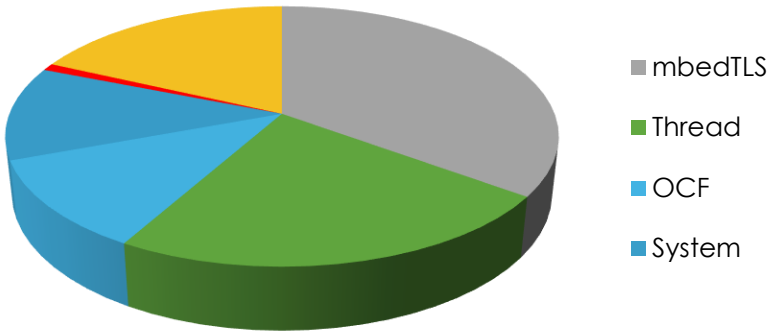


Figure 9 Flash memory usage

There is a fair amount of flash left – just under 93kB. This is enough to develop applications on top of OCF. The biggest single user of flash is mbedTLS, which provides the security function for both OCF and Thread. This is to be expected, because security is a big part of what sets both OCF and Thread apart from less secure connectivity standards. Configuring Thread as an ED frees up an additional 52kB of flash, which can be used by the application if required.

OCF Application Resources

The system being profiled contains a simple module application using up 218 bytes of RAM and just over 4kB of flash. Compared to the rest of the system, it is not much at all. However, within a deployment, additional software is necessary for device drivers and application logic. Therefore, it is necessary to look at the system as a whole to ensure resources are sufficient.

Simple applications should allocate memory on the stack and the heap, which are shared with other parts of the system. Running the system through the OCF certification test tool (CTT) will provide confidence that the resulting application is unlikely to run out of RAM.

If more memory is required, configuring Thread to act as an ED frees up 10kB of RAM, which are not used by any other part of the system and can therefore be dedicated entirely to the application. Additionally, OCF has been specifically engineered to allocate RAM on behalf of the application. Therefore, applications do not tend to require much additional memory.

Security

The OCF specifications have been developed in close alignment with security legislation and industry requirements. This is the responsibility of the OCF Security Oversight Work Group, and which has incorporated this work within the OCF Security Baseline Overview. In summary, OCF complies with all known IoT baseline security requirements.

Public Key Infrastructure (PKI) is the foundation of secure communications over the Internet. It is used, for instance, to assure users that they can trust the websites used to access internet banking and online shopping.

Hence using PKI as an onboarding mechanism and for identity/role certificates is a must for commercial deployments. Implementing PKI on such an embedded platform in OCF, focused on enabling the hardware cryptographic acceleration functions (encryption/decryption) of the M2351. Enabling this functionality meant that the OCF implementation had to be adapted to facilitate hardware cryptography. The effect of this was a time reduction of a PKI DTLS handshake, which went down from around 30 seconds to 7 seconds. This reduction in time makes the system viable.

Implementations of the OCF specifications are extensively tested by the OCF Conformance Test Tool (CTT). Every OCF certified product must pass the CTT suite, which verifies all mandatory statements within the OCF specifications. In this case, the Chili2 module was subject to 119 test cases, consisting of more than 20,000 OCF requests sent over three to four hours. 60 tests focus exclusively on verifying the security functionality of the module.

The Open Connectivity Foundation has put in place traceable links between security legislation, compliance of the OCF specifications to the legislation and compliance of implementations to these specifications. Since clauses within legislation are mapped to clauses within the OCF Specification, and the CTT verifies conformance to the specifications, implementations are assured to comply with legislation worldwide. Latest information about OCF Security baseline compliance can be found at: OCF Security (openconnectivity.org)

Benefits: Security by Design

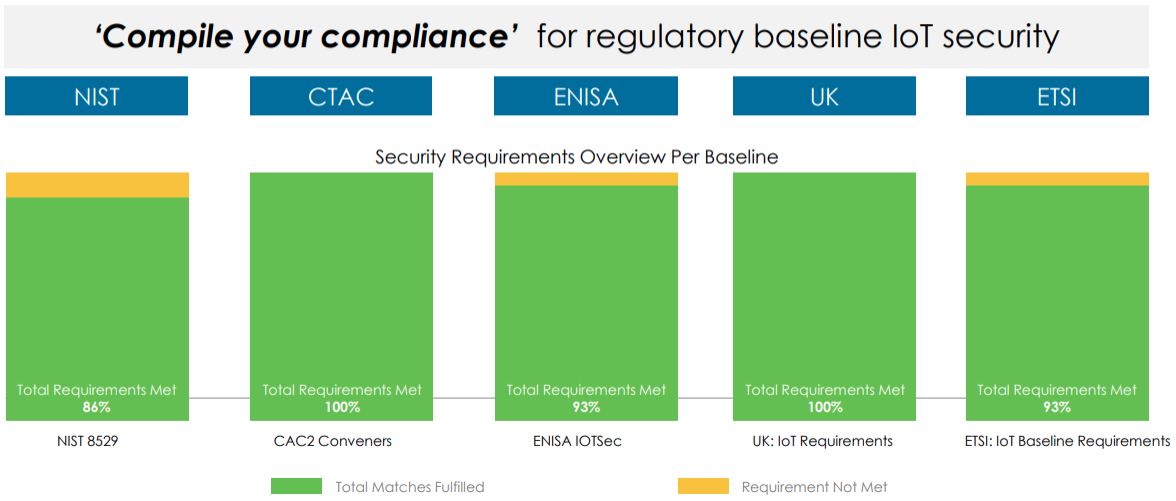


Figure 10 OCF Security baseline compliance

Thread security is based on Password Authenticated Key Exchange by Juggling (J-PAKE). The Thread commissioning flow also has been accelerated by using hardware. This speeds up the Thread commissioning by roughly a factor of ten.

Also, the flash content is encrypted. This means that the application is secured and also all OCF data stored is automatically encrypted. Hence, even if one were to get access to the flash contents, the data on it is secured by using AES-128 encryption.

Conclusion

As a result of this work, Cascoda has now released what it said is the first OCF-certified standards-based low-power IoT module to support both IP (with Thread) PKI security (with OCF). Its certified platform is based on an open-source software development kit (SDK) and comprises the Chili2D module, the first to offer OCF's secure IP framework and application layer, and Thread's low-power and scalable IPv6-based network layer protocol; the required Thread IP router; and OCF cloud connectivity functionality.

The processor used on the Chili2 has only 96kb of RAM, but this is enough to run a REED Thread device and the OCF application stack. It has capacity spare to create additional OCF resources, e.g., add more functionality to the product. The module and the software are available and already being deployed in commercial settings.