PSA Crypto for Silicon Labs Wireless MCUs – Why, What, Where and When

STEVEN COOREMAN AND HENRIK KIRKEBY, NOVEMBER 3, 2020

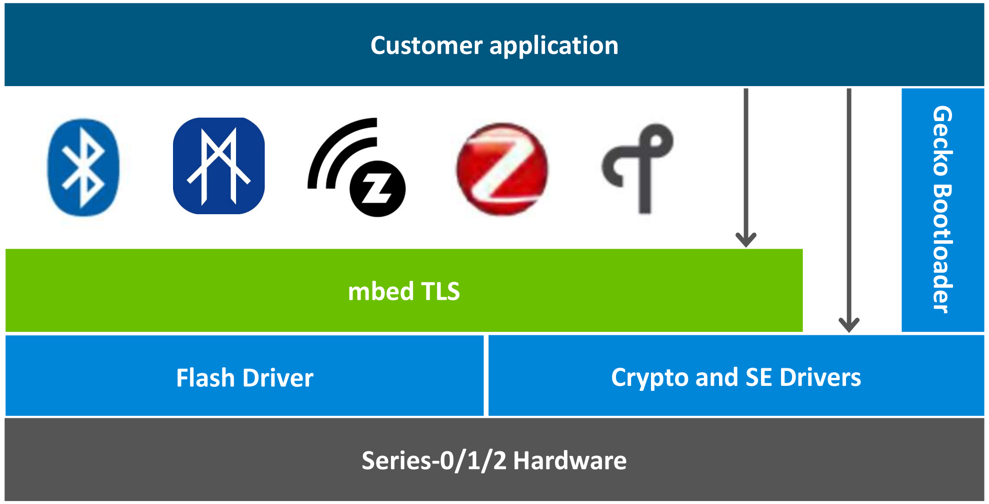
Agenda

- Why
  - Evolution of SL’s wireless MCUs
  - Wireless technology
  - Why PSA Crypto is a good fit
- What
  - Overview of the driver architecture and software stack
  - Migration path
- Where and when
  - SL’s first release of PSA Crypto
  - Remarks on the collaboration model
- Questions

Silicon Labs Wireless MCU Security Evolution

		Series 0	**Series 1**	**Series 2**	
		2010-2013	2013-2018	2018-	
		EFM32xG,EM35x	**xG1, xG1x**	**xG21A, xG22**	**xG21B (Vault)**
AES	Engine speed (128/256-bits)	54/75 cycles	54/75 cycles	22/30 cycles	22/30 cycles
PKI	Engine speed (P-256 sign)	No	~2500k cycles	~350k cycles	~350k cycles
	Autonomous	No	No	Yes	Yes
	Cipher support (bits)	No	P≤256	P≤256	P≤521, Curve25519
Hash	Digest size	No	SHA≤256	SHA≤256	SHA≤512
	Engine speed (SHA-256)	No	66 cycles / 512 bit	66 cycles / 512 bit	66 cycles / 512 bit
AEAD	ChaCha20-Poly1305	No	No	Yes	
Key Protection	DPA countermeasures	No	No	Yes (AES and ECC)	
	Key Isolation	No	No	No	Yes
	Secure key storage	No	No	No	Yes
Identity	Secure identity & attestation	No	No	No	Yes
Boot protect	Secure boot & bootload	Simplistic	GBL	Hardware RoT + GBL	

Wireless Solutions



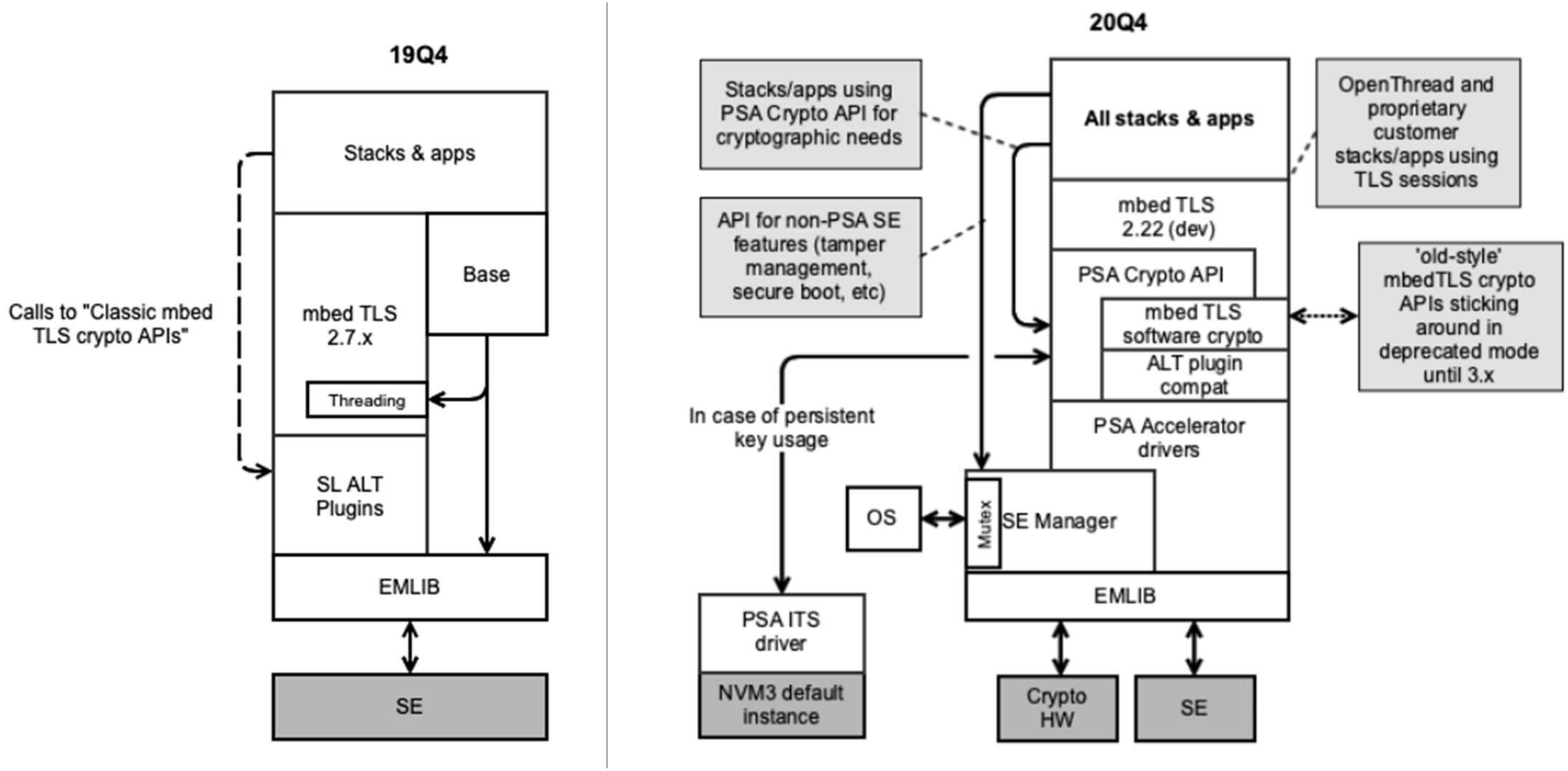
- 5 standard based wireless stack solutions included in the Gecko SDK today, more coming soon
  - Wide range of functional requirements from wireless stacks
- Major business from proprietary wireless
- Low memory footprint applications (<1M flash, <256k RAM)

Why mbed TLS and PSA Crypto is a good fit for Silicon Labs?

- mbed TLS deployed by Silicon Labs SDK since 2015
  - Most required features supported as open-source at the time and more could be added on request
  - Hardware driver model good fit for SL hardware accelerator peripherals
  - Long-term support branches
  - Trustworthy vulnerability incident response process
- However, Series-2 Secure Key Storage not supported by the "classic" mbed TLS APIs
- In 2018, the PSA Crypto API emerged as a viable solution also for Series-2
  - PSA Crypto is a Platform API – offers enablement of legacy hardware accelerators to Series-2 Secure Vault functionality (Secure Key Storage)
  - Formally vetted API and driver interfaces with wide industry acceptance (future proof)
  - mbed TLS 2.2x/3.0 offers a viable upgrade path by introducing the PSA Crypto API alongside the classic API
    - Important because wireless stacks and proprietary solutions may not port at the same time

Driver architecture and migration path

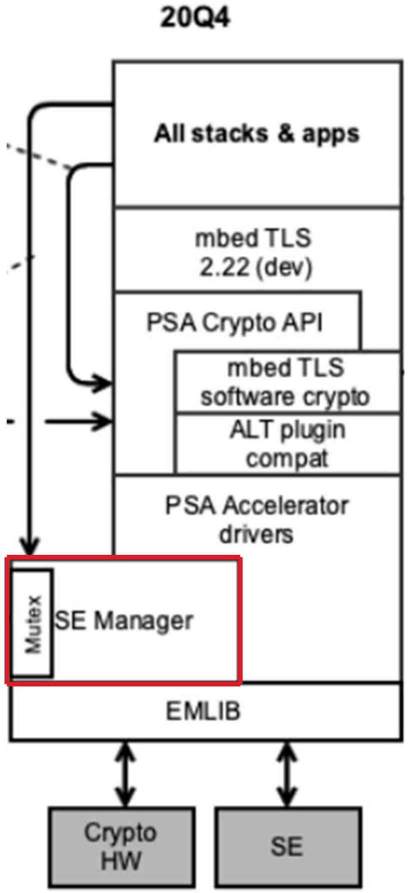
Unification: mbedTLS 2.16 -> mbedTLS 2.21+ w/ PSA Crypto



19Q4
Stacks & apps
Base
Calls to "Classic mbed TLS crypto APIs"
mbed TLS 2.7.x
Threading
SL ALT Plugins
EMLIB
SE
20Q4
Stacks/apps using PSA Crypto API for cryptographic needs
All stacks & apps
OpenThread and proprietary customer stacks/apps using TLS sessions
API for non-PSA SE features (tamper management, secure boot, etc)
mbed TLS 2.22 (dev)
PSA Crypto API
mbed TLS software crypto
ALT plugin compat
'old-style' mbedTLS crypto APIs sticking around in deprecated mode until 3.x
In case of persistent key usage
PSA Accelerator drivers
OS
Mutex
SE Manager
EMLIB
PSA ITS driver
NVM3 default instance
Crypto HW
SE

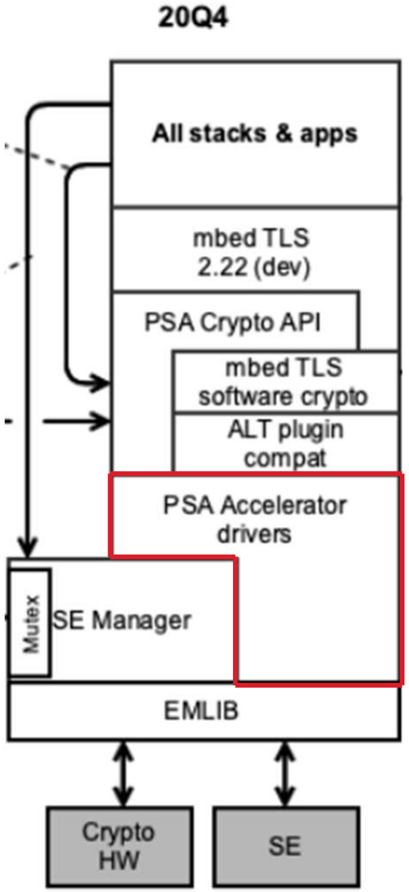
Roll call: SE Manager

- SE Manager is our SE HAL layer predating PSA Crypto
- It provides an interface to the full command set of our (V)SE products
  - Secure boot settings
  - Secure upgrade (host and (V)SE)
  - Secure (remote) unlock
  - Tamper configuration/status
  - Attestation
  - Random Number Generator
  - Device configuration
  - Accelerated cryptography (not on VSE)
  - Key wrapping & management (Vault only)
- SE Manager **is not meant to be** a generic cryptography abstraction
  - It provides nothing more, nothing less than what the hardware is capable of
- SE Manager provides thread-safety at the peripheral-access level when compiled with RTOS support
- SE Manager's APIs for crypto are not considered external APIs
  - Using PSA Crypto for cryptography whenever possible enables fallback scenarios



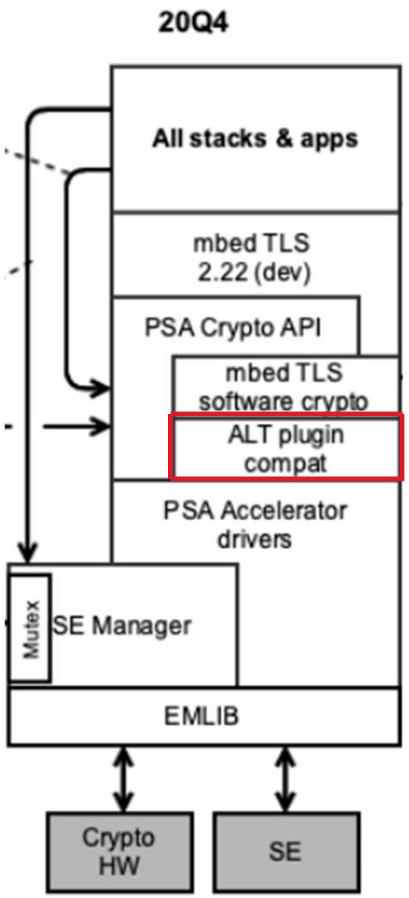
Roll call: PSA Accelerator drivers

- Implement ‘hooks’ the PSA Crypto core can call for accelerating operations
- Implemented for all hardware-backed algorithms
  - Implementations are being made for our supported product families
- If an algorithm is not supported in HW, software fallback is possible
  - This will need to be configured compile-time (default: all fallback turned on)
  - Will be able to turn off fallback to save code space
    - Drop references to the mbedTLS software crypto implementations
  - Mechanism to do this automatically based on application requirements and hardware capabilities is in the works
- ‘Transparent’ drivers are accelerators
  - They get all keys fed to them JIT in plaintext by the PSA Crypto core
- ‘Opaque’ drivers are secure elements providing key storage/wrapping
  - Once a key is wrapped by or stored inside of a secure element, it is opaque
  - An opaque driver can also offer transparent functionality through dual-driver use



Roll call: *_ALT compatibility layer

- Provides a migration path for those not able or willing to move towards using the PSA APIs immediately
- Implements the old-style mbedTLS acceleration hooks on top of the PSA Accelerator drivers for SL hardware
  - PSA accelerator drivers are our focus, and what we support going forward
  - Reduced duplication by having *_ALT on top of PSA accelerators
  - Slight drop in performance
    - Effect can be dampened by multi-file compilation / LTO
- One should be able to swap out the 20Q2 mbedTLS folder with the 20Q4 one, and expect everything to continue to work
  - Same config file results in the same feature set
  - Slight change in file set for compilation (file addition/removal from upstream)
    - Not an issue specific to this migration



PSA APIs vs mbedTLS – porting isn't hard!

- PSA Cryptography APIdoc: https://armmbed.github.io/mbed-crypto/html/index.html
- PSA Crypto getting started: https://github.com/ARMmbed/mbedtls/blob/development/docs/getting_started.md
  - Ignore where it says 'mbed crypto' – this is about the PSA Cryptography functional API
- PSA APIs are grouped by algorithm category
  - The exact algorithm is a parameter to the function, not an individual function
  - When porting, suggest to hardcode this to make multifile compilation / LTO work optimally
- PSA APIs don't take key input directly
  - Keys need to be imported before use
  - APIs that need key input take a key identifier
- PSA APIs exist in both streaming and single-shot modes
  - For supported algorithm categories
- PSA APIs always return `psa_status_t`

Implications

- Standardised use of buffers
  - Input buffers:
    - Pointer
    - Length of input data
  - Output/inout buffers:
    - Pointer
    - Length of allocated buffer (to avoid buffer overflow)
    - size_t output pointer (to indicate how much data was written into the buffer)
- Standardised use of context structures
  - All context structures are as large as the largest structure within the algorithm family
- Opaque structures when running the operation through a driver
  - Driver-specific contexts get allocated dynamically, meaning dynamic memory is now a requirement for all
  - No specific structure init/free function
    - Init = zero-allocate
    - Free = abort

Timelines and challenges

Where and When

- Release date for Gecko SDK 3.1 with PSA Crypto is December 9
- The release will be made available through Simplicity Studio available from www.silabs.com

Remarks on the collaboration model

- Open-source collaborative model fits well with mbed TLS’ value proposition
- The PSA Crypto project roadmap depends heavily on contributions (unknown X factor)
- What is needed to deliver on the roadmap?
  - More contributions from the industry
  - Complete specification work
  - More reviewer and maintainer bandwidth
  - Transform CI system to a fully open system (remove dependency on ARM internal CI systems)

Thank you! Questions?

STEVEN COOREMAN AND HENRIK KIRKEBY, NOVEMBER 3, 2020