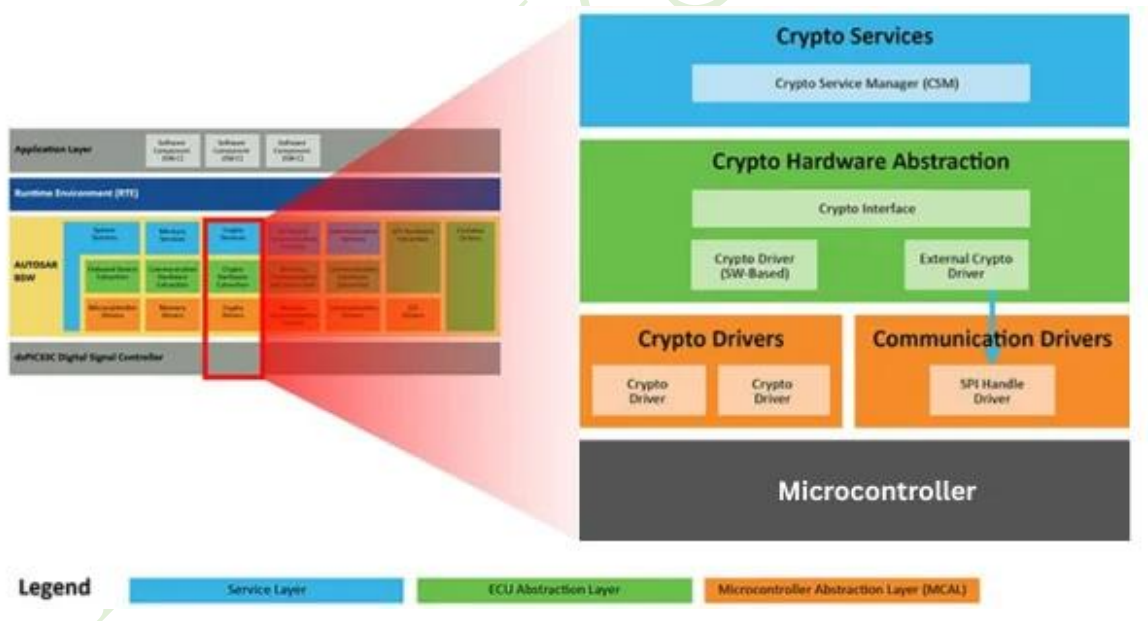




AUTOSAR Crypto Stack Syllabus



Week 1: AUTOSAR Security Fundamentals**Theory**

- ❖ Role of security in automotive systems (ISO 21434, ISO 26262)
- ❖ Position of Crypto Stack in AUTOSAR layered architecture
- ❖ Threat models: replay, tampering, unauthorized ECU access

Practical Configuration & Testing

- ❖ Create a baseline ECU configuration project
 - ❖ Map Crypto Stack into the BSW architecture
 - ❖ Test with simple AES + SHA software functions
-

Week 2: Cryptography Basics Refresher**Theory**

- ❖ Symmetric vs Asymmetric crypto in AUTOSAR context
- ❖ Hashes, MACs, Digital Signatures, RNG requirements
- ❖ Security objectives in Crypto Stack: confidentiality, integrity, authenticity

Practical Configuration & Testing

- ❖ Define crypto primitives (AES, SHA, HMAC) in config
 - ❖ Run test vectors for AES + SHA via generated stubs
 - ❖ Validate deterministic results vs expected vectors
-

Week 3: AUTOSAR Crypto Stack Architecture**Theory**

- ❖ Crypto Service Manager (CSM), Crypto Interface (CryIf), Crypto Driver
- ❖ AUTOSAR Job Model: Jobs, Primitives, Queues, Channels
- ❖ Error detection and Diagnostic Event Trace (DET)

Practical Configuration & Testing

- ❖ Configure ECUC containers for Crypto modules (CSM, CryIf, Driver)
 - ❖ Generate code and inspect headers/APIs
 - ❖ Test job submission for AES + SHA
-

Week 4: Key Management in AUTOSAR**Theory**

- ❖ Key lifecycle: creation, storage, update, invalidation, revocation
- ❖ Key sets and key elements in AUTOSAR configuration
- ❖ Handling asymmetric keys and certificates

Practical Configuration & Testing

- ❖ Configure key sets in ECUC
 - ❖ Generate CSM Key APIs (KeyElementSet, KeySetValid, KeyInvalidate)
 - ❖ Test key load/update/invalidate flow in ECU
-

Week 5: Crypto Service Manager (CSM) Basics**Theory**

- ❖ CSM APIs: Init, Encrypt, Decrypt, Hash, Sign, Verify, MAC, Random
- ❖ Synchronous vs Asynchronous job handling
- ❖ Job lifecycle and callbacks

Practical Configuration & Testing

- ❖ Add AES & Hash jobs to CSM config
 - ❖ Generate code and call Csm_Hash, Csm_Encrypt from application
 - ❖ Validate callbacks and error reporting
-

Week 6: CSM Advanced Topics**Theory**

- ❖ CSM Key APIs in detail (KeyElementSet, KeyGet, KeyInvalidate)
- ❖ Job cancellation and concurrency handling
- ❖ Notification callbacks and error propagation

Practical Configuration & Testing

- ❖ Configure multi-job queues in CSM
 - ❖ Generate code for multiple job types
 - ❖ Test job cancellation, error reporting
-

Week 7: Crypto Interface (CryIf)**Theory**

- ❖ Purpose of CryIf as middleware
- ❖ Routing between multiple drivers
- ❖ Error mapping and callback forwarding

Practical Configuration & Testing

- ❖ Configure CryIf channels for AES + SHA + RSA
 - ❖ Generate mapping tables between CSM and Crypto Drivers
 - ❖ Run a test job routed through CryIf and check results
-

Week 8: Crypto Driver Fundamentals**Theory**

- ❖ Role of Crypto Driver in AUTOSAR stack
- ❖ APIs: Init, Encrypt, Decrypt, Hash, RNG, Sign, Verify
- ❖ Handling job queues and key material inside driver

Practical Configuration & Testing

- ❖ Configure a software Crypto Driver for AES + SHA
 - ❖ Generate driver configuration code
 - ❖ Execute AES job directly via driver APIs
-

Week 9: Advanced Crypto Driver Use**Theory**

- ❖ Support for asymmetric cryptography (RSA/ECC)
- ❖ Handling key formats (binary, ASN.1, internal storage)
- ❖ Error handling & reporting at driver level

Practical Configuration & Testing

- ❖ Configure driver for RSA/ECC signature + verification jobs
 - ❖ Generate APIs and integrate with CryIf
 - ❖ Test asymmetric key pair signing & verification
-

Week 10: Security Use Cases Integration**Theory**

- ❖ Secure Onboard Communication (SecOC)
- ❖ Secure Diagnostics (UDS services 0x27, 0x29)
- ❖ Secure Boot and firmware validation
- ❖ OTA update security

Practical Configuration & Testing

- ❖ Configure CSM jobs for SecOC MAC generation
 - ❖ Generate UDS authentication challenge-response config
 - ❖ Test firmware validation flow using signature verification
-

Week 11: Testing & Performance Validation**Theory**

- ❖ Unit testing of crypto jobs
- ❖ Integration testing across CSM + CryIf + Driver
- ❖ Performance measurement (latency, throughput, memory footprint)
- ❖ Fault injection and robustness validation

Practical Configuration & Testing

- ❖ Run functional tests with generated code (AES, SHA, RSA)
 - ❖ Perform invalid key/error input tests
 - ❖ Measure performance metrics (CPU, RAM usage)
-

Week 12: Capstone Project – Secure ECU Communication**Project Goal**

Configure, generate, and test a complete AUTOSAR Crypto Stack system for secure ECU-to-ECU communication.

Requirements

- ❖ Full CSM + CrylIf + Crypto Driver configuration
- ❖ Implement:
 - Encrypted + authenticated message exchange
 - Freshness management with counters/nonces
 - UDS authentication flow
 - Secure Boot validation flow
- ❖ Generate AUTOSAR-compliant code and validate via test cases

Deliverables

- ❖ ECUC configuration files
- ❖ Generated code + application integration
- ❖ Test logs (functional + negative + performance)
- ❖ Final project demo + presentation

☑ Outcome for Students

By the end of this 3-Month AUTOSAR Crypto Stack Training, students will:

- Configure AUTOSAR Crypto modules (CSM, CrylIf, CryptoDriver)
- Generate AUTOSAR-compliant code for crypto jobs and key management
- Test encryption, MAC, signatures, key handling, random number generation
- Integrate crypto with SecOC, UDS, Secure Boot, OTA
- Deliver a working Secure ECU Communication System that passes AUTOSAR test flows