Intel's Common Data Security Architecture

Presented to the OpenGroup

December 11, 1996

Why A Common Data Security Architecture?

- Early 1995, we recognized the need for an *open*, *interoperable*, *cross platform*, security infrastructure
- Our idea: define a basic infrastructure so the primary ingredients of any security solution are "just there"
  - interface specs designed to remove interoperability hurdles
  - lower TTM hurdles to incorporate security features in legacy and new products
  - openly reviewed to ensure robustness
- Intel has developed this functionality for use in its own products (e.g., LANDesk Management)

CDSA Design Goals

- Create an *open*, *cross platform*, *interoperable*, core security infrastructure
- Support use and management of the fundamental elements of security:
  - certificates
  - trust
  - cryptography
  - integrity
- Support multiple programming environments
  - C and JAVA
- Make extensible above and below
- Build to be operating system independent

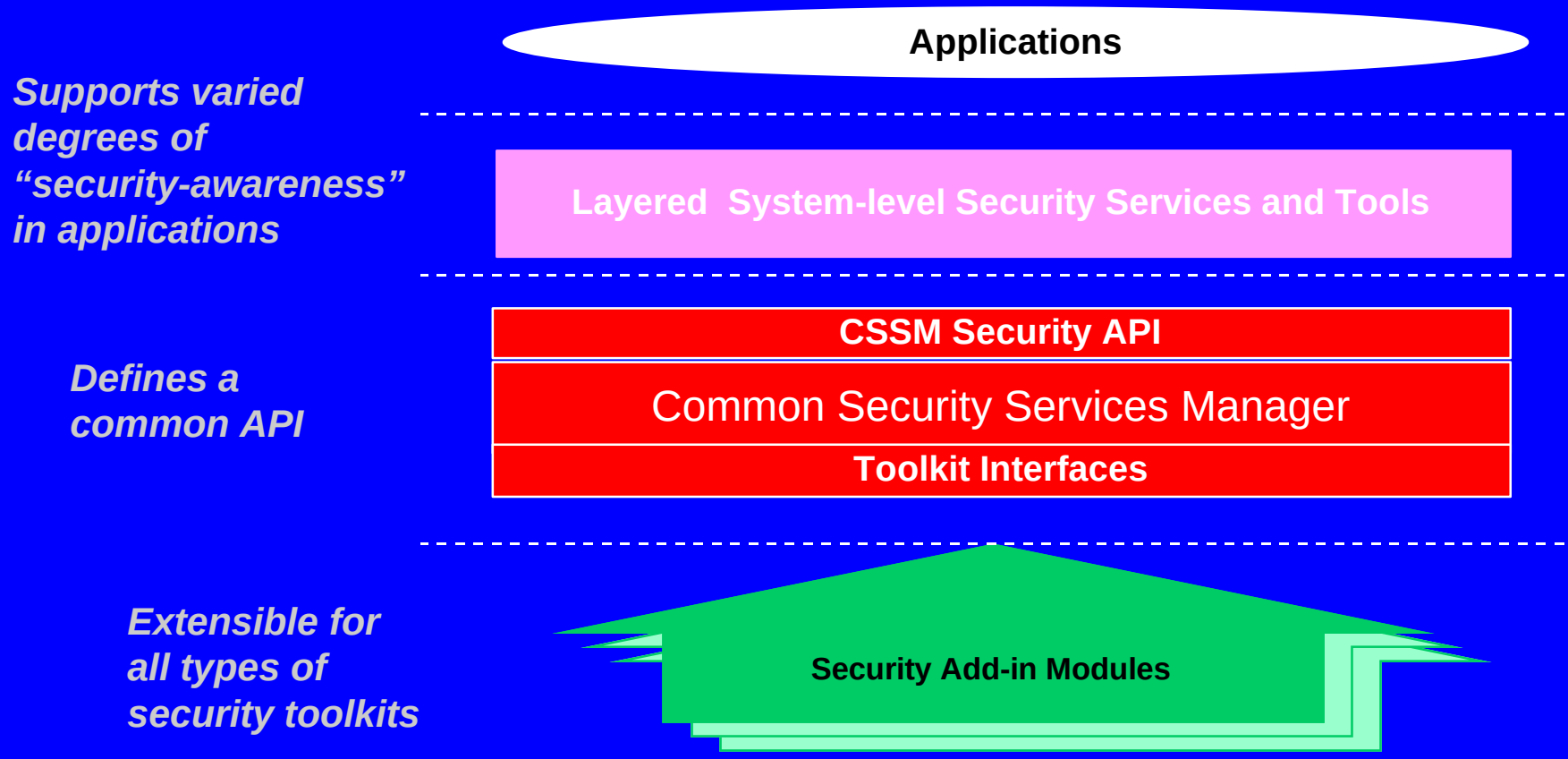
CDSA Status

- We published an early version of the specifications at the 1996 RSA Conference
- On the Intel Web site (http://www.intel.com/ial/security/)
  - evolutionary versions of the specifications for review/comment
  - reference implementation for Windows* 95 and Windows NT*

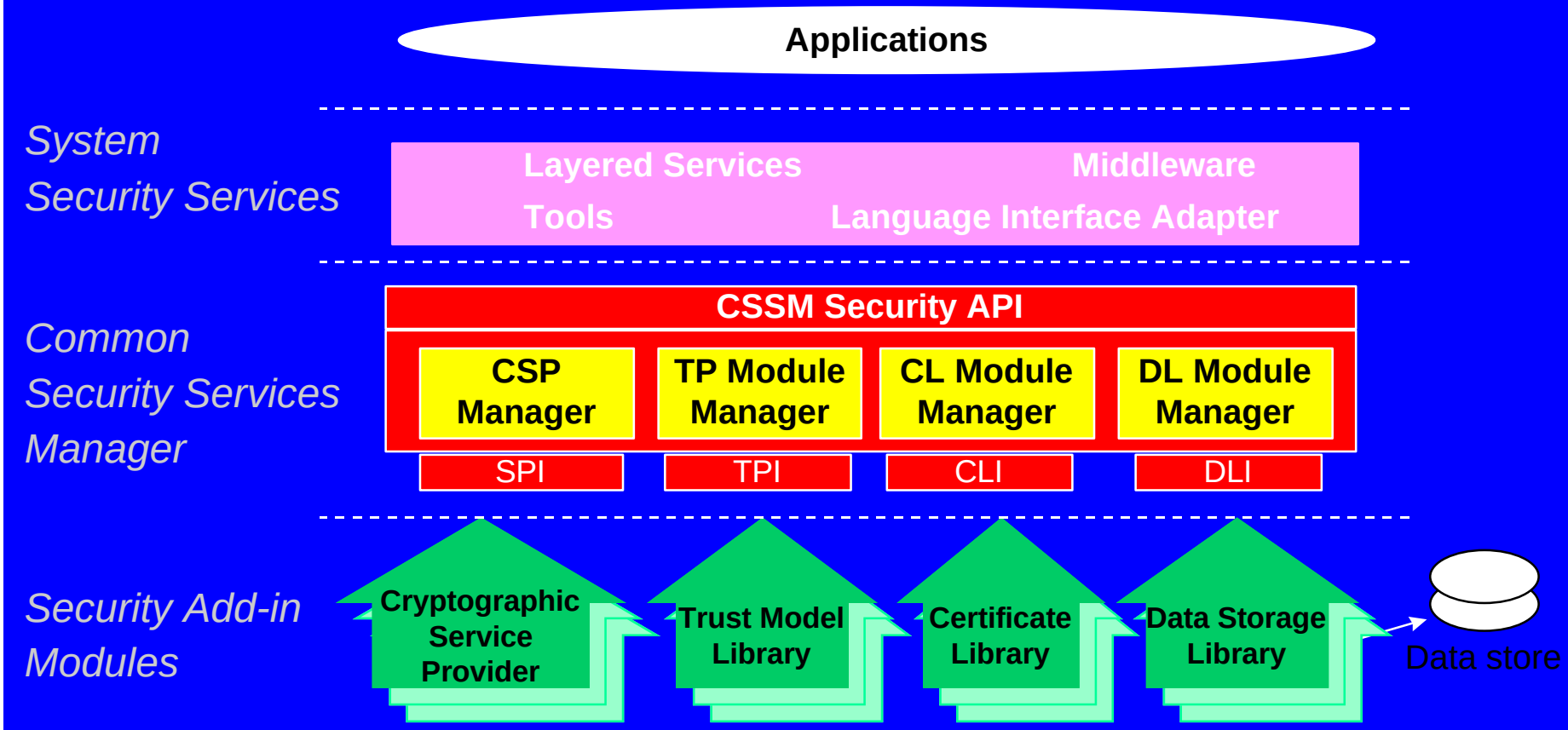
Technology Topics

- CDSA Overview
- CSSM Infrastructure
  - Management services
- Add-in Security Service Modules
  - Trust Policy, Certificate Library and Data Storage Modules
- Application Usage Models
- CSSM Java API

CDSA - a four layer architecture



Common Security Services Manager (CSSM)

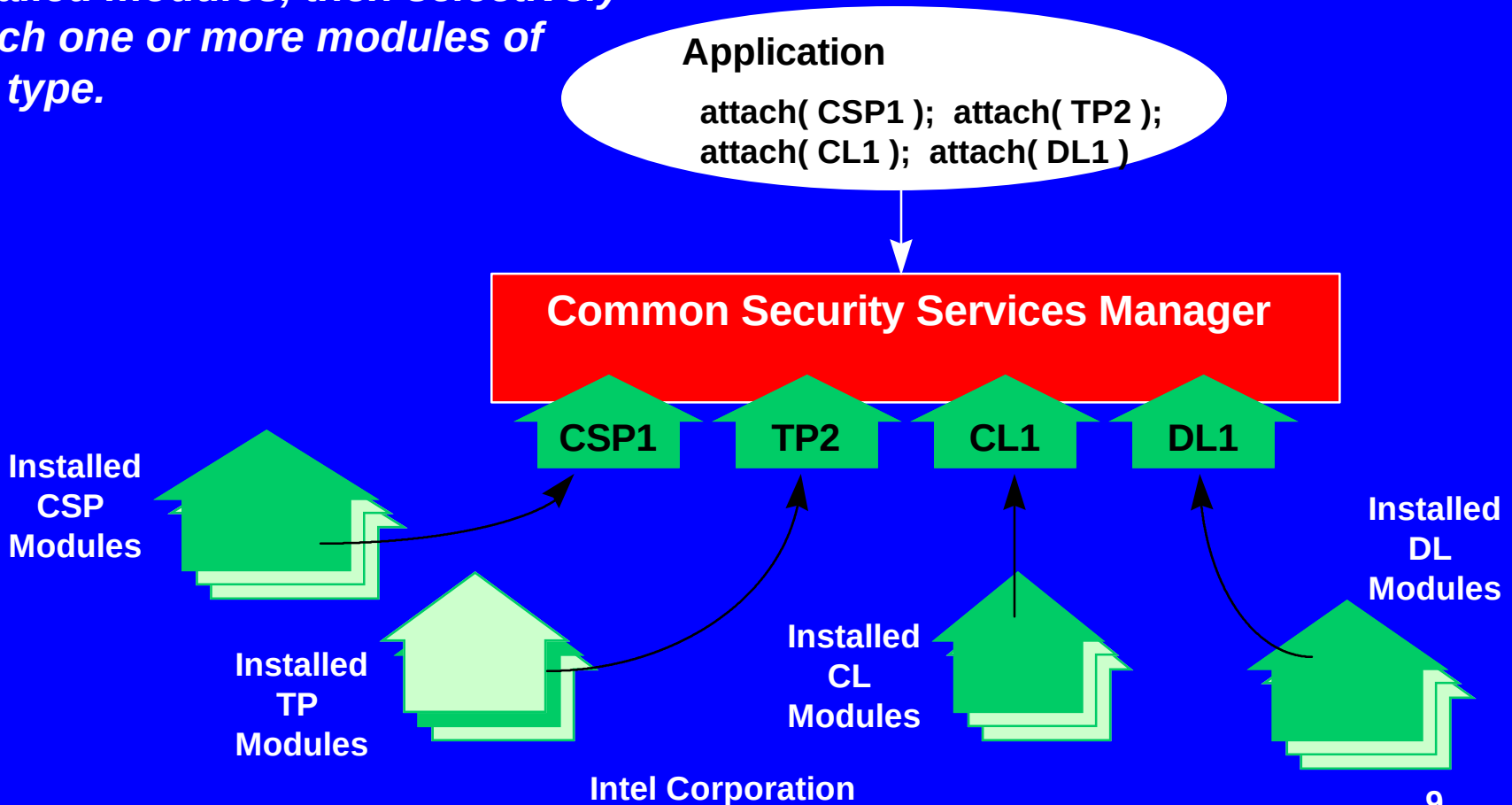


Security Add-in Modules

- Trust Policy Modules
  - semantics of a trust model
- Certificate Library Modules
  - syntactic, memory-based manipulation of certs
- Data Storage Library Modules
  - syntactic, persistence of certificates
- Cryptographic Services Modules
  - cryptographic operations in software or hardware
- Modules may perform I/O, network ops, etc.
- Multiple modules of each type
  - supports an open market for toolkit vendors

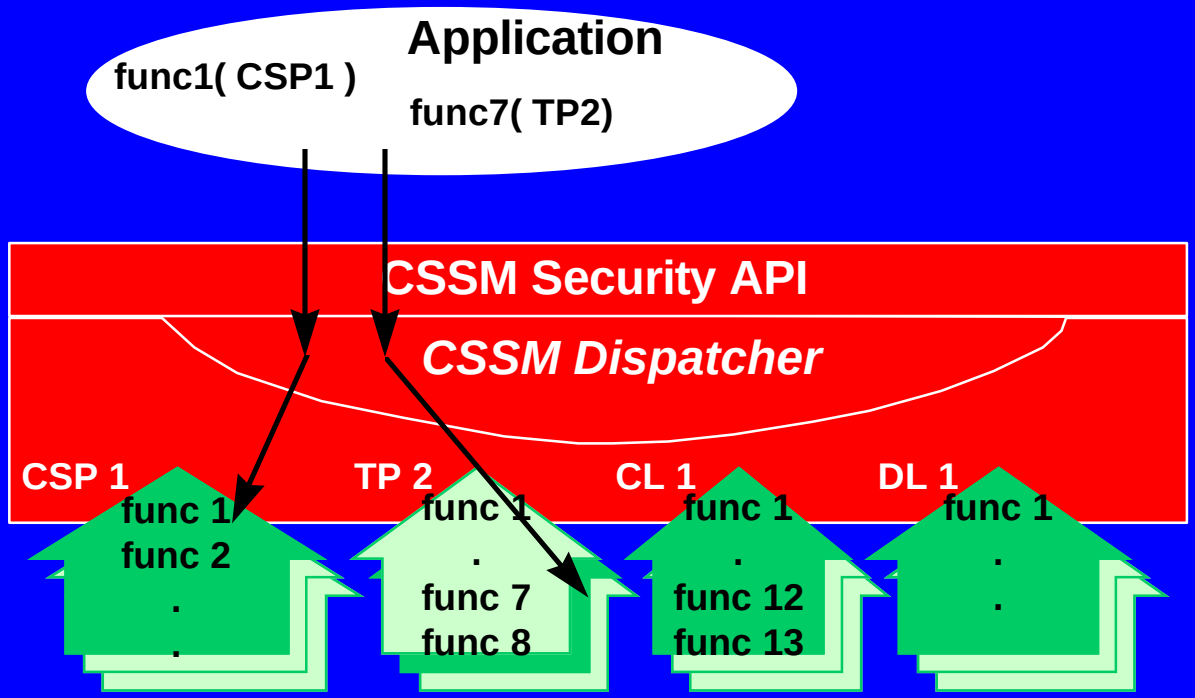
Applications Select Security Service Modules

Step 1: Applications query for installed modules, then selectively attach one or more modules of any type.



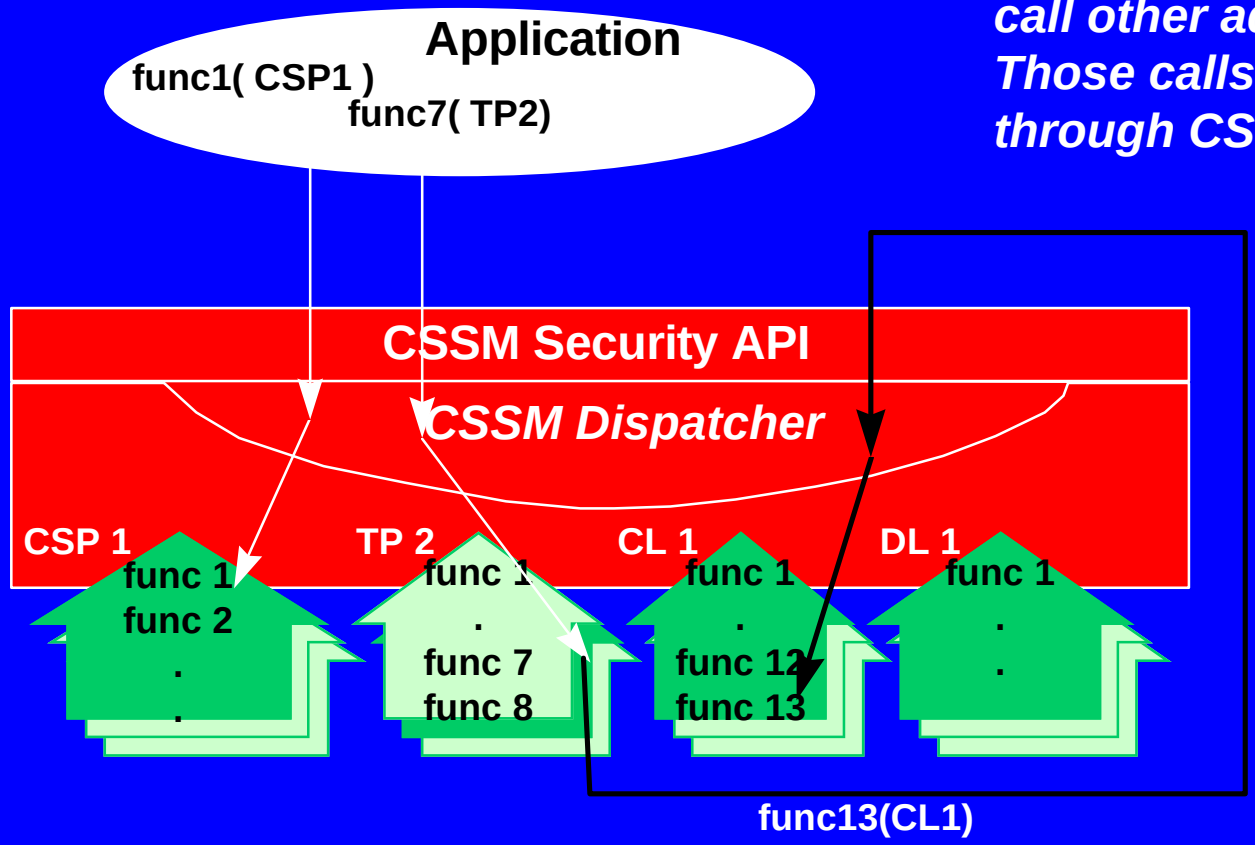
Applications Invoke Security Services

Step 2: Applications call the CSSM security API.
Calls are selectively dispatched to security service modules.



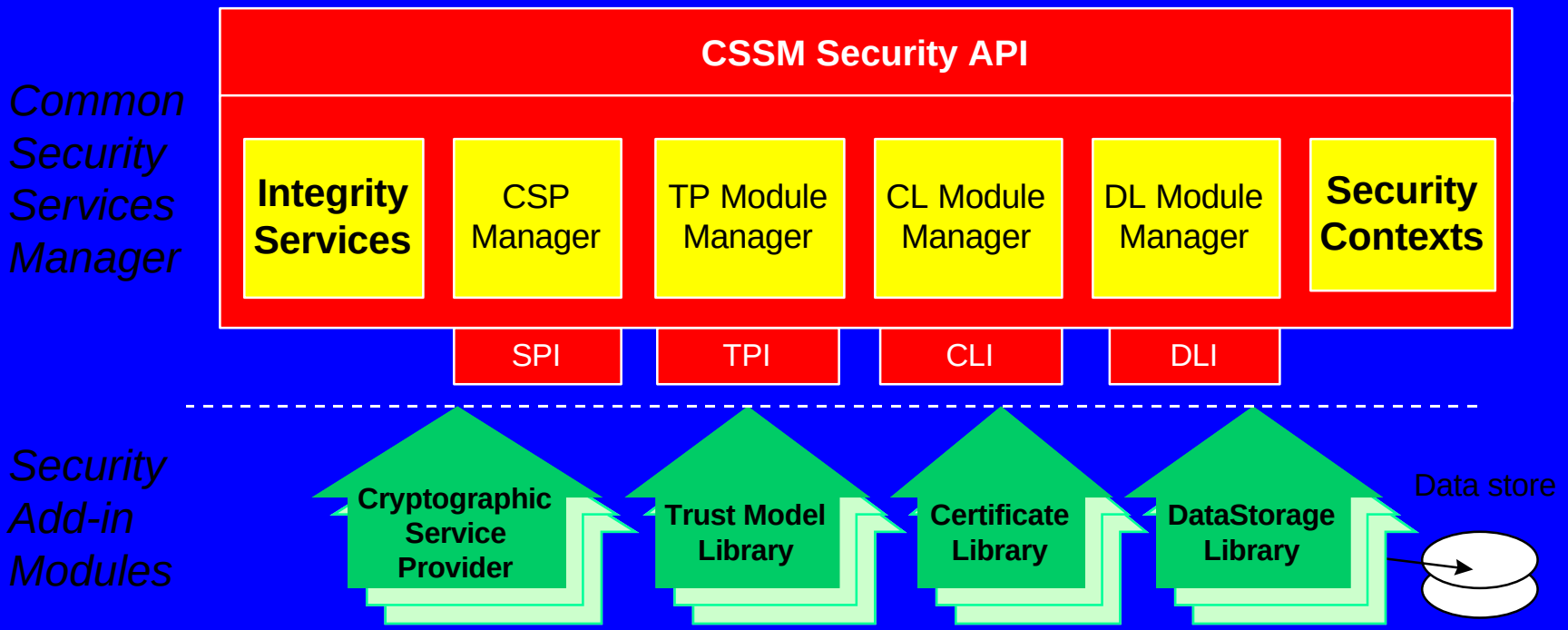
A General Dispatching Mechanism

Add-in security modules can call other add-in modules. Those calls are dispatched through CSSM.



Additional CSSM Services

Security Context Management and Self-integrity Checking

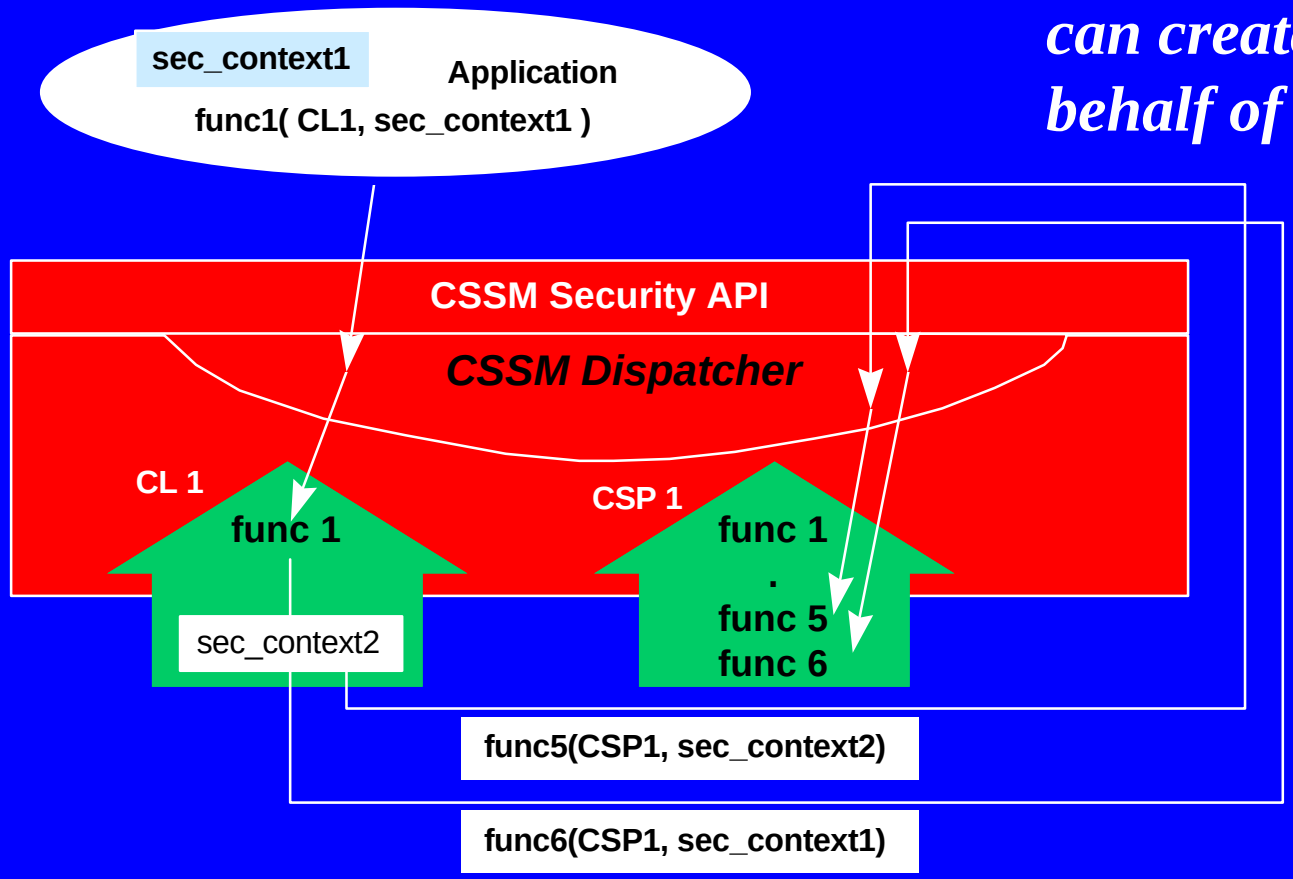


Security Contexts

- Stateful Information
  - for re-entrancy
  - for multi-threading
- Application creates one or more contexts
  - explicit and implicit
- API defines contexts by type
  - key_exchange, signature, symmetric, digest, MAC, random_number, asymmetric, key_generation, generic
- Info used by Cert libs, CSPs, Storage libs

Security Context

Add-in security modules can create contexts on behalf of an application.



Trust Policy API

- Access to certificate-based trust models
  - semantics of trust
- Generic API supports different trust models
  - from hierarchical to introducer
- Basic categories of operations:
  - sign, verify, revoke certificates and CRLs
  - verify application-specific action

Trust Policy APIs

TP_ CertVerify ( )
TP_ CertSign ( )
TP_ CertRevoke ( )

TP_ CrlVerify ( )
TP_ CrlSign ( )
TP_ApplyCrl To Db ( )

TP_VerifyAction ( )
TP_PassThrough ( )

Certificate Management API

- Defines memory-based manipulation of
  - certificates
  - certificate revocation lists (CRL)
- Generic API so libraries can support different Certificate types
- Uses OIDs for naming fields in certificates and CRLs
- Basic categories of operations:
  - create, sign, verify
  - view, get_field_values
  - type_translations

Certificate Lib APIs for Certs

CL_ CertCreate ( )
CL_ CertSign ( )
CL_ CertUsign ( )
CL_ CertVerify ( )

CL_ CertDescribeFormat ( )
CL_ CertView ( )
CL_ CertGetFirstFieldValue ( )
CL_ CertGetNextFieldValue ( )
CL_ CertAbortQuery ( )
CL_ CertGetKeyinfo ( )
CL_ CertGetAllFields ( )

CL_CertImport ( )
CL_ CertExport ( )

CL_PassThrough ( )

What's a PassThrough Function?

- An open, generic function for add-in modules to provide unique services
- PassThrough(cl_id, msg_id, inputs, results)
  - Msg_id identifies the requested function
  - inputs are a list of input data values
  - results are a list of return values
- Behaviors documented in the module guide
  - used by knowledgable applications
- Example:
  - performance enhancement that retrieves only the v1 portion of an X.509 v3 cert

Certificate Lib APIs for CRLs

CL_ CrlCreate ( )
CL_ CrlAddCert ( )
CL_ CrlRemoveCert ( )

CL_ CrlSign ( )
CL_ CrlVerify ( )

CL_ IsCertInCrl ( )

CL_ CrlDescribeFormat ( )
CL_ CrlGetFirstFieldValue ( )
CL_ CrlGetNextFieldValue( )
CL_ CrlAbortQuery( )

Data Storage API

- Persistence for
  - certificates
  - certificate revocation lists (CRL)
- Persistence is orthogonal to memory-based manipulation
- API supports a range of implementations
  - full data base management systems
  - file systems
  - schema-based versus blob-based
- API defines store mgmt, cert & CRL ops

APIs for Data Store Management

DL_DbOpen ( )

DL_ DbClose ( )

DL_ DbCreate ( )

DL_ DbDelete ( )

DL_ DbImport ( )

DL_ DbExport ( )

Storage Lib APIs for Certificates

DL_ CertInsert ( )
DL_ CertDelete ( )

DL_ CertRevoke ( )

DL_ CertGetFirst ( )
DL_ CertGetNext ( )
DL_ CertAbortQuery ( )

DL_PassThrough ( )

Storage Lib APIs for CRLs

DL_ Crl Insert ( )

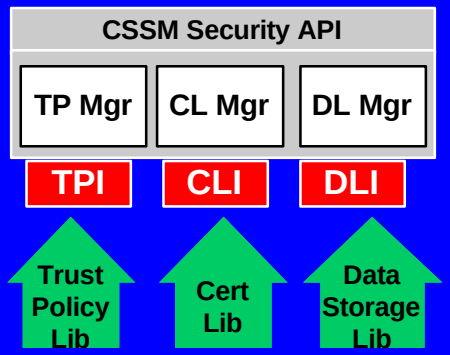
DL_ CrlDelete ( )

DL_ CrlGetFisrt ( )

DL_ CrlGetNext ( )

DL_ CrlAbortQuery ( )

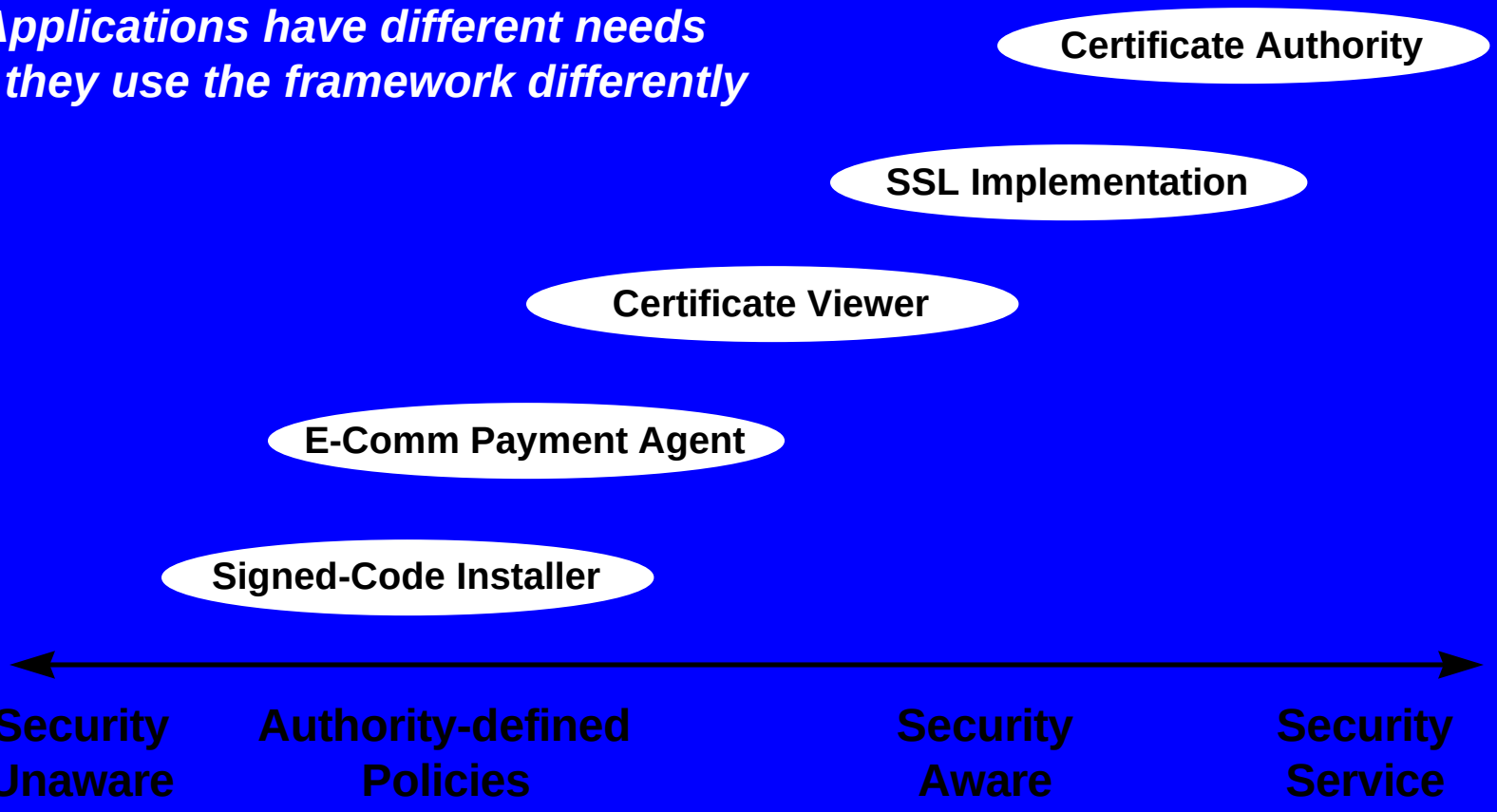
Interfaces (TPI, CLI and DLI) for the Toolkit Provider



- Two categories of functions
  - module management
  - services available to the application
- Module Management
  - install, uninstall, attach, detach, describe
- Services
  - one-to-one with the API functions
  - may or may not implement a given function
  - option to provide functions via pass-through

Application Usage Models

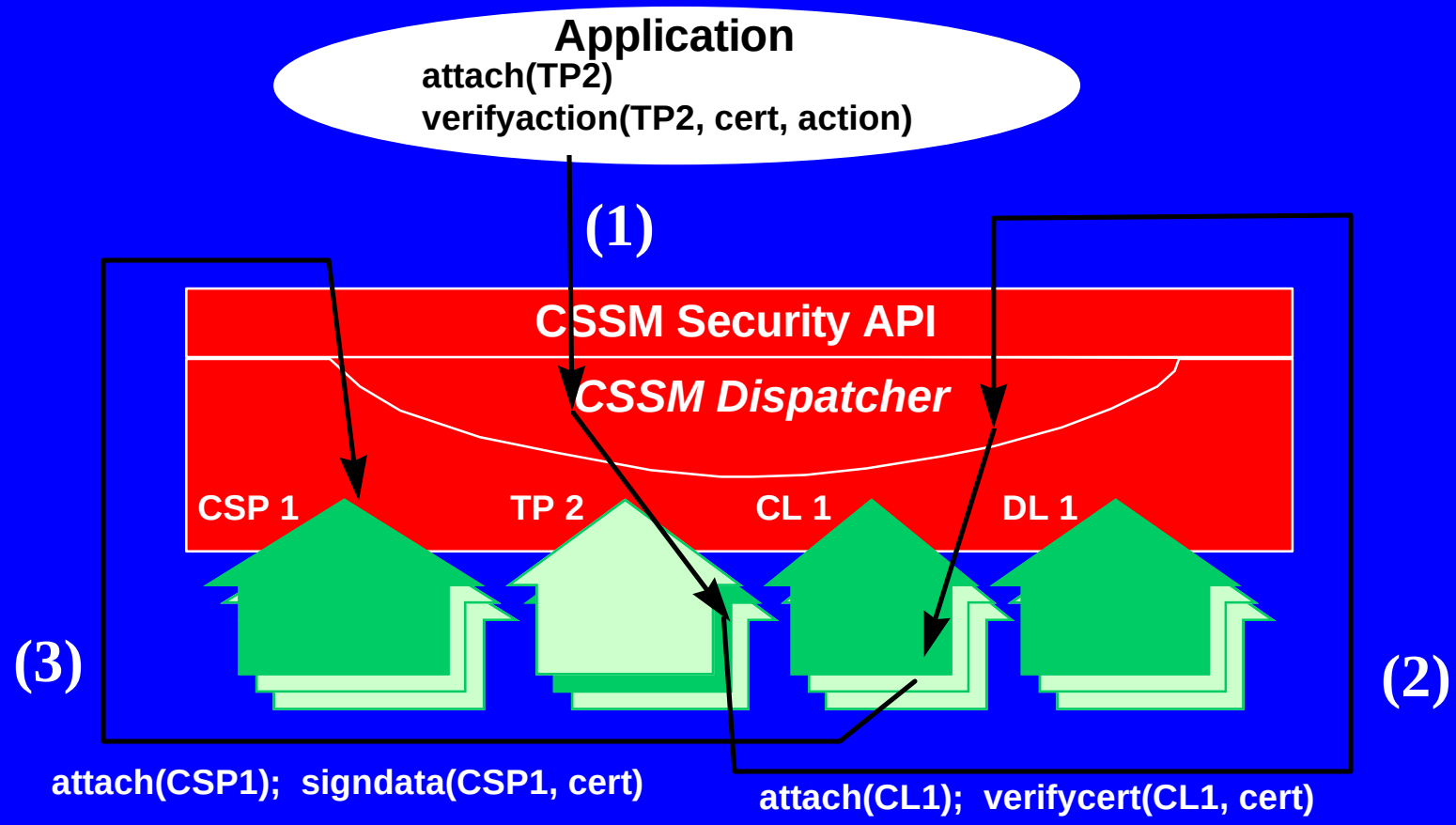
Applications have different needs
- they use the framework differently



Authority-controlled Security Policies

- Application
  - attaches a Trust Policy
  - calls VerifyAction(TP, certificate, action)
- Trust Policy
  - attaches and calls other modules to perform syntactic manipulation of certificates and CRLs
    - » verify identity, integrity, and authorization according to the implemented policy.
  - does not duplicate other modules (great re-use)
- Application sees only the trust policy

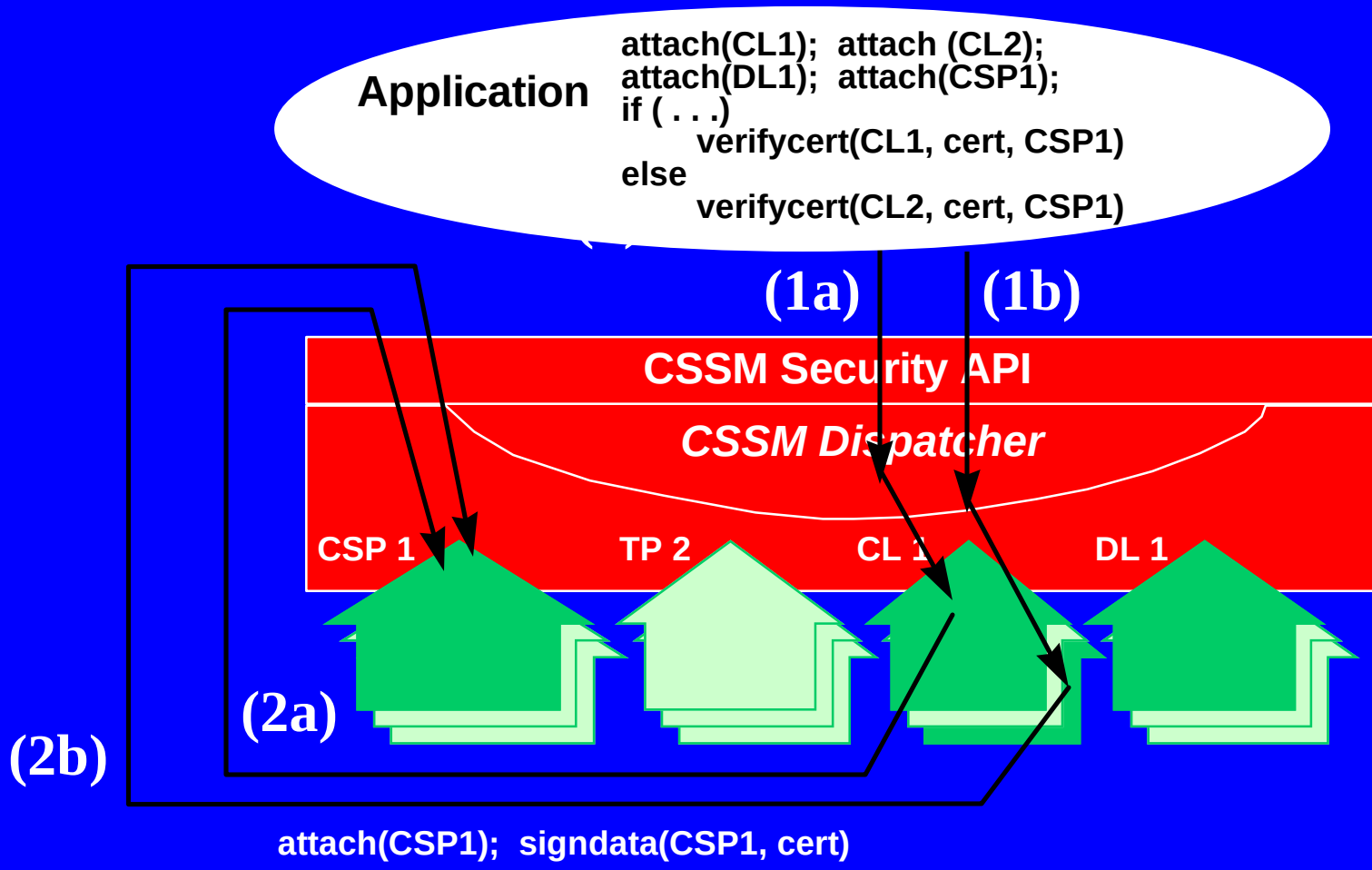
Execution Flow for Authority-controlled Policies



Security-aware Applications with Self-contained Policy Elements

- Application
  - may or may not use a Trust Policy
  - calls Cert Libs, Data Storage Libs, & CSPs directly
- Cert Libs, Data Storage Libs, & CSPs
  - perform syntactic manipulation of certificates & CRLs
    - » verify identity, integrity, and authorization as directed by the application.
- Application sees a set of modules and understands the capabilities of those modules (including pass-through operations)

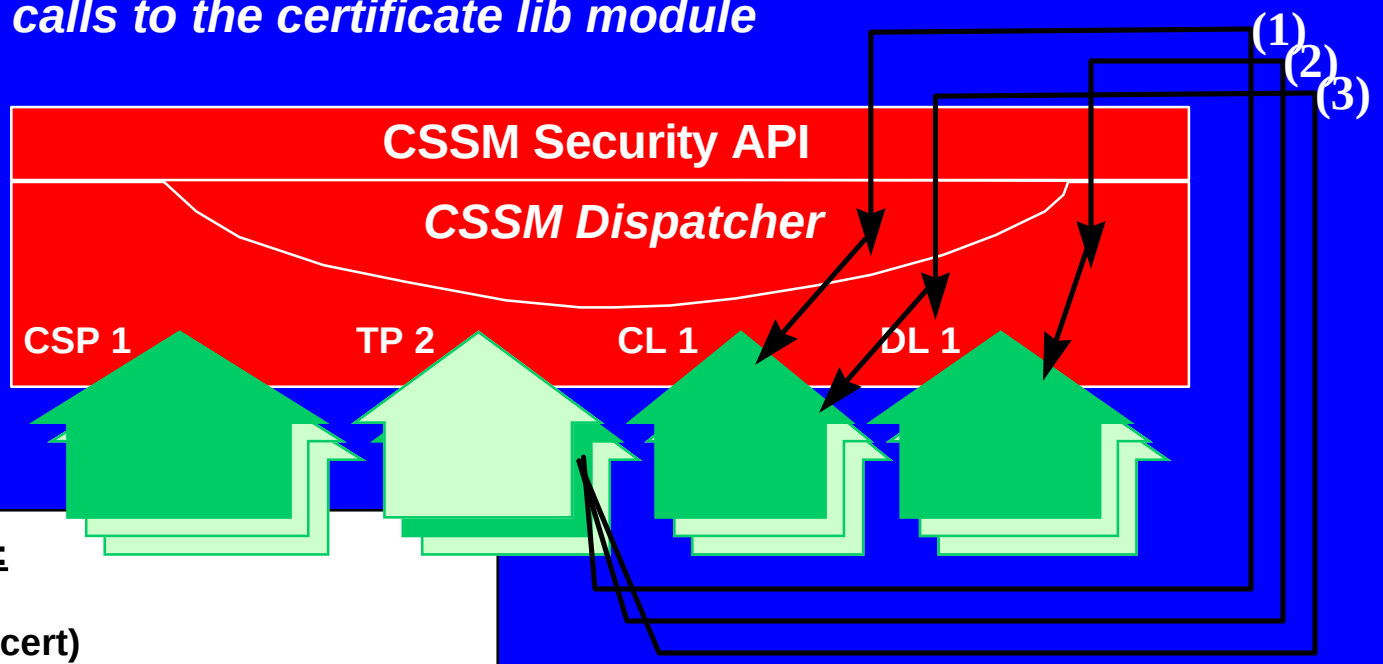
Execution Flow for Security-aware Applications



Trust Hierarchies and Certificate Chains

Walking Certificate Chains

- the trust policy walks the chain by repeated calls to the certificate lib module



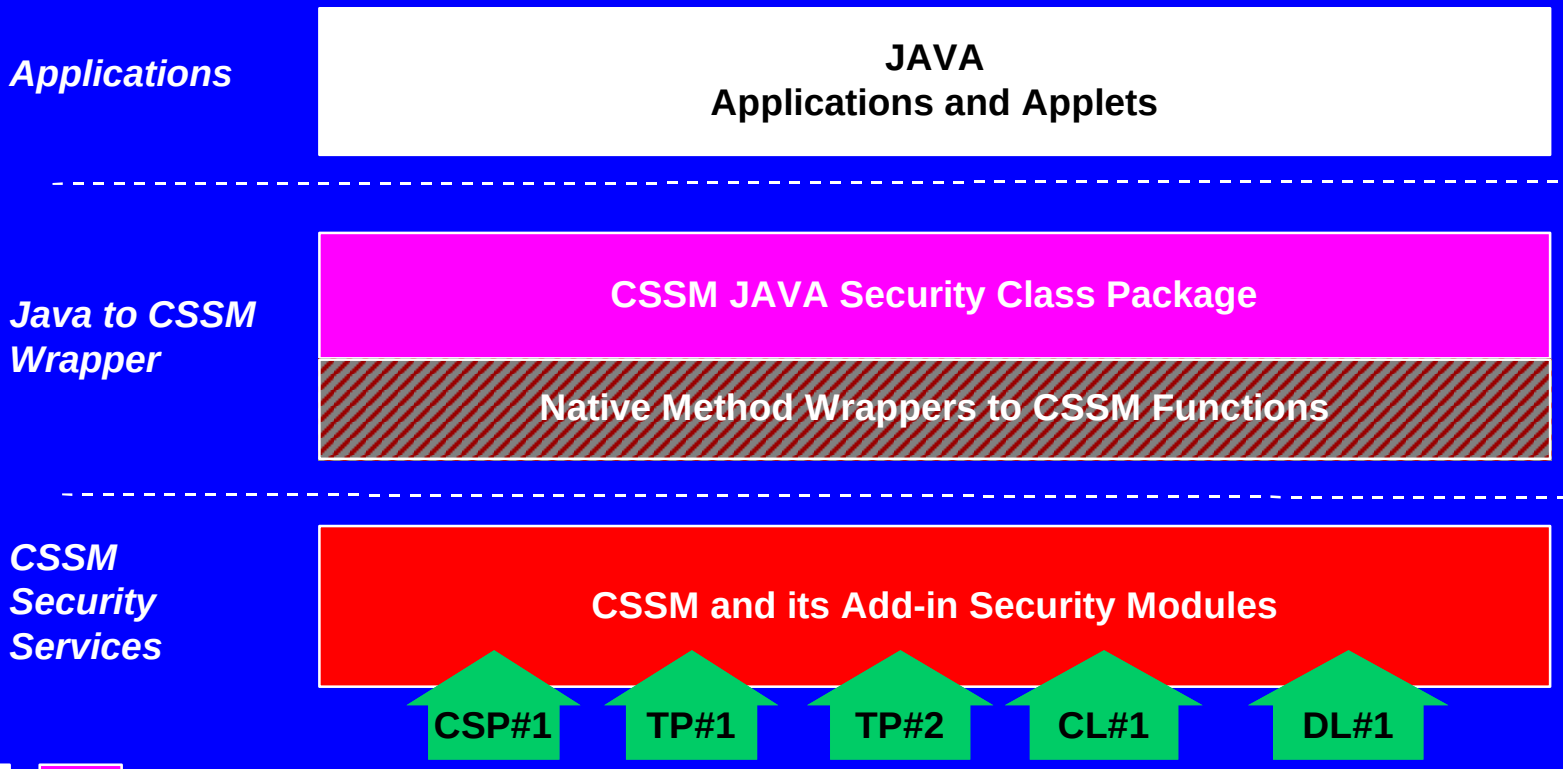
Trust policy code:

```
attach(CL1)
verifycert(CL1, cert)
certgetfirstfieldvalue(CL1, cert, issuer-OID)
attach (DL1))
nextcert = certgetfirst(Issuer-OID)
verifycert(CL1, nextcert)
```

CSSM Concepts in Java

- Goal:
  - make the same security concepts/objects accessible to the Java programmer
  - facilitate programmers working in C and Java
  - one security infrastructure implementation on the platform
- Approach
  - define a set of Java classes based on CSSM
  - implement class methods as native methods that invoke CSSM's C language API

Java Support



12 Java Security Classes

- Certificate
- CRL
- Crypto
- DataStore
- Key
- SecurityContext
- TrustPolicy
- CLRegistry
- CSPRegistry
- DLRegistry
- TPRegistry
- CSSMException

Summary

- CDSA is an open, extensible, security architecture suitable for multiple platforms
- CDSA supports
  - security-aware to security-unaware apps
  - development of a wide range of trust models
  - enhanced products from toolkit vendors
- Reference implementation is available
  - http://www.intel.com/ial/security