The Impact of the Common Data Security Architecture (CDSA) on Telemetry Post Processing Architectures

Jeffrey R. Kalibjian
CounterSign Software, Inc.

ABSTRACT

It is an increasing requirement that commercial satellite telemetry data product be protected from unauthorized access during transmission to ground stations. While the technology (cryptography) to secure telemetry data product is well known, the software infrastructure to support such security is costly, and very customized. Further, many software packages have difficulty interoperating. The Common Data Security Architecture [1] [2] [3] (originally proposed by the Intel Corporation, and now adopted by the Open Group), is a set of common cryptographic [4] and public key infrastructure (PKI) application programming interfaces (APIs) which will facilitate better cryptographic interoperability as well as making cryptographic resources more readily available in telemetry post processing environments.

KEY WORDS

cryptography, public key infrastructure (PKI), Common Data Security Architecture (CDSA)

INTRODUCTION

The Common Data Security Architecture is a set of programming application interfaces (APIs) which specify cryptographic, certificate, trust, and data storage operations. The specification allows for the functionality implied by the APIs to be provided by "add-in" modules which can be implemented by third party vendors. After describing the CDSA architecture and discussing its implications for security applications, examples will be given illustrating how the technology may be used positively impact telemetry post processing operations.

ELEMENTS OF CDSA ARCHITECTURE

The central concept behind CDSA architecture is that common low-level security operations can be generalized into programmatic APIs that are applicable across computing platforms. With this approach, security applications can be written once to utilize the CDSA APIs; and recompiled when appropriate for deployment on target computer systems (as long as there are implementations of CDSA on those target systems). The programmatic APIs, together with the infrastructure to utilize them, are commonly referred to as the CDSA Framework. It is important to realize that the Framework itself cannot accomplish anything. They are merely APIs which aggregate data. Actual functionality is achieved when so called “add-in” libraries are attached to the Framework to implement the functionality implied by the APIs (Figure 1).

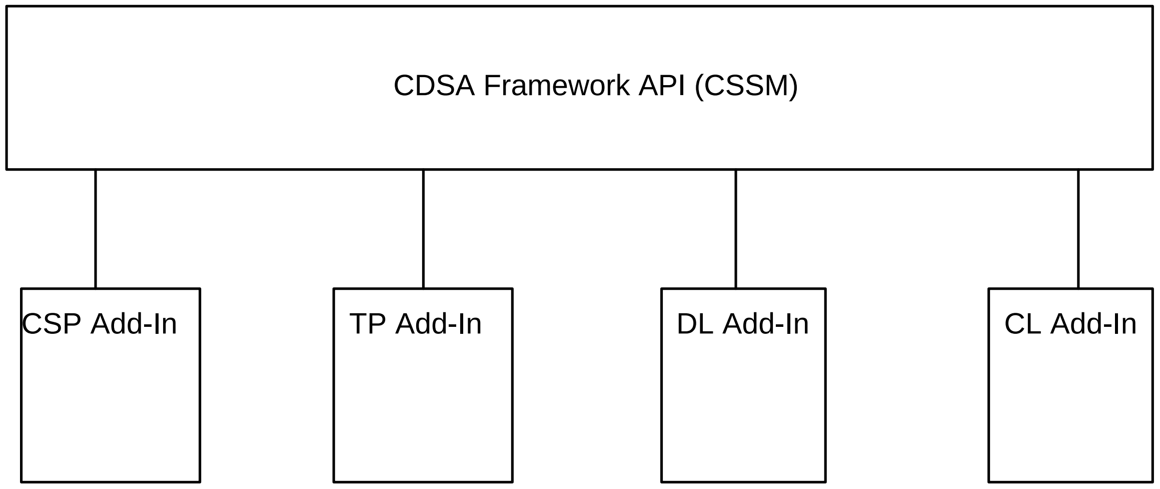

Figure 1. CDSA Architecture.

It is important to realize that add-ins are platform dependent. That is, they are specifically implemented to function on a particular operating system. In addition, whereas just as source code written to the CDSA APIs is platform portable; actual cryptographic objects (e.g. keys, encrypted data, signatures, etc.) created by calls to those APIs will also be typically interoperable with other objects created by CDSA add-ins running on other operating systems-----as long as those add-ins adhere to accepted industry standards for lower level cryptographic information exchange; e.g. PKCS 1, X509, etc. Finally, it should be noted that add-ins may be implemented in hardware as well as software.

Appendix 1 describes the typical operations implemented by the CSP and CL add-ins. Very briefly, the Cryptographic Service Provider (CSP) add-in does what its name implies-----implement general cryptographic algorithms (e.g. symmetric and asymmetric algorithms, digital signature, etc.). The Certificate Library (CL) add-in implements digital identity functions by providing functionality for embedding cryptographic keys in certificate structures. Facilities are also provided for certificate verification and certificate revocation. The Trust Policy (TP) library provides for functionality that can evaluate a user's privileges based on logic and their digital identity certificate (hence the term "trust policy" add-in). Finally, the Data Library (DL) implements a database storage facility for objects which are created by CDSA add-in modules (e.g. keys, certificates, etc.)

ADD-IN VALIDATION

The sensitive nature of applications utilizing CDSA require that they be able to confirm that the add-ins being loaded by the application have not be tampered with by outside hostile third parties. To insure this, the CDSA architecture implements an elaborate checking policy to insure the integrity of add-ins and the Framework. All add-ins have associated with them a separate manifest file which contains a digitally signed hash of the associated add-in library. Before the CDSA Framework loads a requested add-in, the Framework calculates a hash on the add-in library. This hash value is compared with the signed hash value in the associated manifest file. If the hashes match (and the signature on the hash in the manifest is verified), the add-in library will be loaded. However, before control is returned to the application, the add-in (now in computer memory) calculates a hash over itself in memory and again compares the result to the value in the associated manifest file. This is known as the "self check." It insures that the bits that were actually loaded were indeed the bits that were originally validated on disk. After the "self check," the add-in attempts to confirm that the CDSA Framework (that is the software that loaded the add-in) has not been tampered with. This is done by hashing the Framework software in memory and comparing the results to the signed Framework manifest. If these hashes compare, and the add-in can confirm that the calling function that loaded it returns to the region of memory where the Framework hash was performed, the application can be fairly well assured that the add-in it has just loaded (and the Framework it is using) has not been tampered with by a third party. The validation architecture is illustrated in Figure 2.

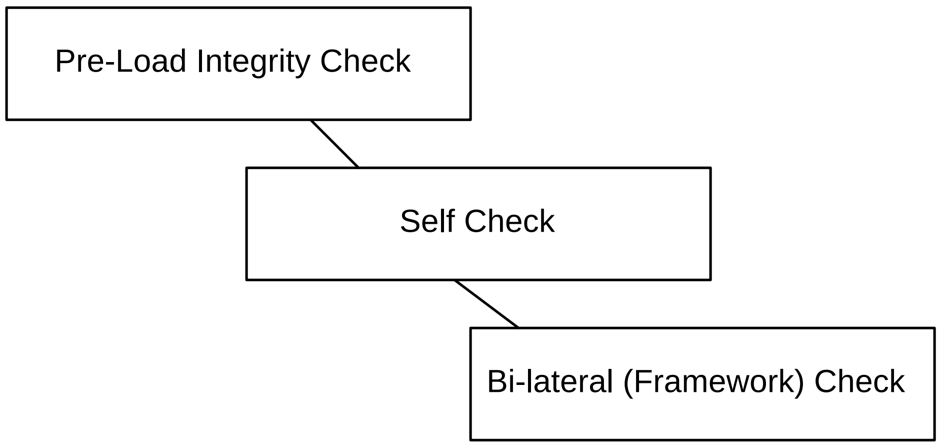

Figure 2. CDSA Integrity Validation Architecture.

CDSA AND TELEMETRY POST PROCESSING

The widespread adoption of CDSA technology will bring great benefits to the development of telemetry post processing applications which must operate on an encrypted data product. First, post processing applications written to the CDSA API will not have to be re-coded to deploy on new operating systems. Thus, the source code for these application will become more portable. Today there are a myriad of crypto libraries available on many different operating systems. Unfortunately, all these libraries utilize different APIs. Thus, if a post processing application needs to be moved to a different operating system, on which a currently used crypto library is not available, that post processing application will need to modify all its crypto API's in order to use the new crypto library on the new target operating system.. Shortly, CDSA technology will be offered on all major computer operating systems. This is already occurring in the marketplace. CDSA today is offered under HP-UX 11.0 [5], and very shortly Apple and IBM will begin to offer CDSA services on newer versions of their operating systems.

Second, telemetry post processing applications which are capable of processing encrypted data product should become more prevalent in the marketplace. Currently any company commercially selling an application utilizing a cryptographic library must pay royalties for the use of the library in their product. This practice has served to inhibit creation of general security products; and has no doubt also effected the telemetry post processing area as well. The companies currently offering CDSA in their operating systems, are choosing to allow developers to utilize the crypto services in finished products royalty free.

Finally, third, CDSA's other capabilities can be used to help facilitate secure telemetry data management. This can include encryption of post processed data results, public/private key storage, certificate creation, and implementation of trust policies to manage access of raw and post processed data.

CONCLUSION

CDSA is a complete infrastructure for cryptographic, certificate, trust and data storage operations. Its adoption by the Open Group insures that it will become the defacto standard for use with security applications. This will positively impact telemetry post processing architectures by allowing them to develop facilities for handling secure data product in a cost effective manner.

Appendix 1

General CDSA 1.2 CSP and CL Functionality

CSP (Cryptographic Operations)

Symmetric Key Generation
Asymmetric Key Generation
Symmetric Encryption/Decryption
Asymmetric Encryption/Decryption
Digesting
Digital signature/Digital signature verification
Generate/Verify MAC
Random Number Generation
Key Wrap/Unwrap

CL (Certificate Operations)

Certificate Verification
CRL Verification
Certificate Generation
Certificate Parsing/Manipulation
CRL Parsing/Manipulation
Certificate Signing
CRL Signing

References

[1] Intel Corporation,"Common Data Security Architecture Specification, Version 1.2," March 1997, http://www.intel.com/ial/security/specs_1_2.htm.

[2] Intel Corporation,"Common Security Services Manager, Version 1.2," March 1997, http://www.intel.com/ial/security/specs_1_2.htm

[3] Sargent, Richard, "CDSA Explained," The Open Group, 1998, ISBN 1-85912-231-0.

[4] Schneier, Bruce,"Applied Cryptography: Protocols, Algorithms, and Source Code in C," Second edition, New York, John Wiley and Sons, Inc., 1996, ISBN 0-471-12845-7.

[5] The Hewlett-Packard Corporation, "HP Praesidium Common Data Security Architecture (CDSA 1.2) White Paper," June 1999, http://www.software.hp.com.