

Practical and Secure Trust Anchor Management and Usage

Carl Wallace

Cygnacom Solutions

7925 Jones Branch Drive Suite 5200

McLean, VA 22102

cwallace@cygnacom.com

Geoff Beier

Cygnacom Solutions

7925 Jones Branch Drive Suite 5200

McLean, VA 22102

gbeier@cygnacom.com

ABSTRACT

Public Key Infrastructure (PKI) security depends upon secure management and usage of trust anchors. Unfortunately, widely used mechanisms, management models and usage practices related to trust anchors undermine security and impede flexibility. In this paper, we identify problems with existing mechanisms, discuss emerging standards and describe a solution that integrates with some widely used applications.

Categories and Subject Descriptors

K.6.5 [Management of Computing and Information Systems]: Security and Protection - *authentication*.

General Terms

Security

Keywords

Trust anchor management, public key infrastructure (PKI).

1. INTRODUCTION

Trust anchors (TAs) are used for a variety of purposes. For example, trust anchors are used when a web browser authenticates a web server, when an email client verifies a signature on an email message or prepares an encrypted email message and when a domain controller authenticates a user logging in with a smart card. In short, a TA is used whenever a PKI is securely used. Trust Anchor Management Requirements [6] provides the following definition for a TA:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IDtrust '10, April 13–15, 2010, Gaithersburg, Maryland, U.S.A.
Copyright© 2010 ACM 978-1-60558-895-7/10/04...\$10.00.

“A Trust Anchor is a public key and associated data used by a relying party to validate a signature on a signed object where the object is either:

- *a public key certificate that begins a certification path terminated by a signature certificate or encryption certificate*
- *an object, other than a public key certificate or certificate revocation list (CRL), that cannot be validated via use of a certification path.”*

Trust Anchor Management Requirements [6] also provides a definition for a trust anchor store:

“A trust anchor store is a set of one or more trust anchors stored in a device. A trust anchor store may be managed by one or more trust anchor managers. A device may have more than one trust anchor store, each of which may be used by one or more applications.”

In current practice, a trust anchor is a (typically self-signed) certificate that resides in a trust anchor store. Despite their importance, trust anchor stores are usually managed, to a large extent, by software vendors. Trust anchor store users have few or no enforceable constraints available to limit the extent of trust accorded to the trust anchors in the trust anchor store or to the software vendor managing the trust anchor store.

This paper briefly describes current trust anchor management tools and practices, identifies some problems with the status quo and describes an implementation that provides alternative trust anchor management mechanisms for applications that use the Microsoft Crypto API (CAPI) certification path processing interfaces.

2. Current Trust Anchor Management and Usage

In most common scenarios, trust anchors are distributed and managed by operating system and application vendors. TA stores are initialized during software installation and are often changed by software updates. Proprietary operating system-specific or application-specific tools are used to customize trust anchor store contents. These actions may be undone, however, by automated trust anchor store updates or routine software updates.

Synchronization of trust anchor stores from different vendors (or even the same vendor) requires manual steps using proprietary tools. Comparison of trust anchor store contents is a similarly manual affair.

Most operating systems and applications use certificates to represent trust anchor information. In some cases, a collection of trust anchors may be represented using a “certificates only” Cryptographic Message Syntax (CMS) SignedData message. Some applications may require distinguished encoding rules (DER) encoded certificates or privacy enhanced mail (PEM) encoded certificates, but this is a fairly minor problem as conversion tools are readily available.

The following sections provide an overview of some widely used mechanisms and discuss the primary problems with these mechanisms.

2.1 Overview of selected current mechanisms

2.1.1 Microsoft Windows

Many applications that operate on Microsoft Windows platforms use the trust anchor stores built into the operating system. A variety of interfaces are available for adding trust anchors to a trust anchor store, including the following:

- Right-clicking a certificate file, choosing “Install Certificate” from the resulting menu and selecting a trust anchor store destination,
- Installing a certificate into a trust anchor store using the Microsoft Management Console (MMC),
- Installing a certificate into a trust anchor store using an application-provided interface, such as Internet Explorer (IE),
- Installing a certificate into a trust anchor store using group policy or System Center Configuration Manager (SCCM).

The MMC interface to the trust anchor store is shown below.

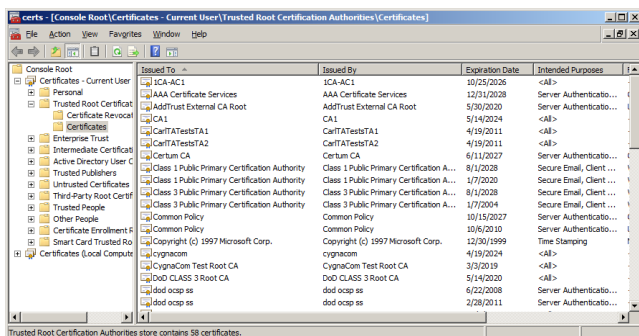


Figure 1 MMC view of a trust anchor store

Some options, such as MMC and Internet Explorer, allow for the specification of certain trust anchor constraints, which are referred

to in the user interface as properties or purposes. Constraints are configured using a dialog like the one shown below in Figure 2. The constraint options are very similar to extended key usage values, with a difference being that extended key usage extensions are not processed across a certification path but the constraints configured here appear to be. On Windows Vista SP 2 systems, there are 38 purposes available for selection. When a trust anchor is manually installed, all purposes are enabled by default.

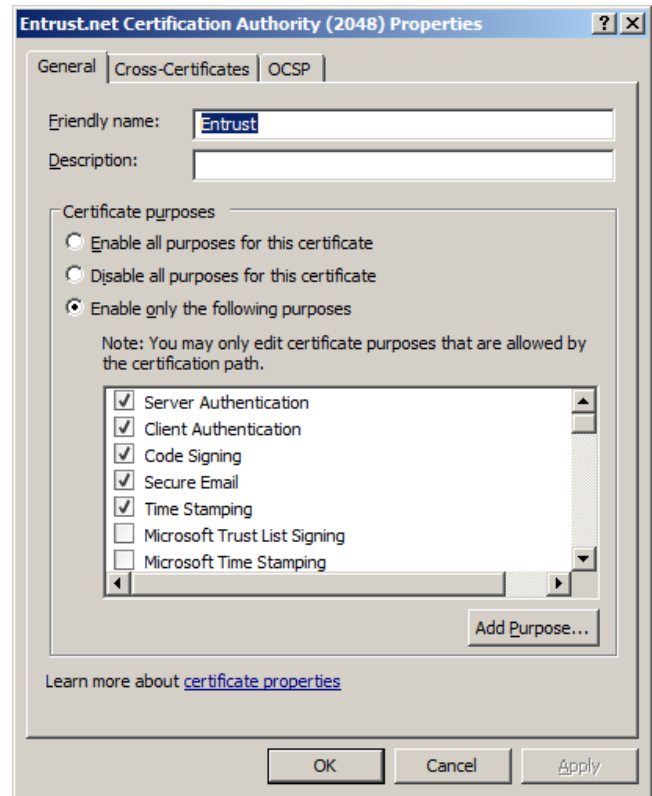


Figure 2 Microsoft trust anchor constraints dialog

In addition to manual trust anchor installation, Windows provides automatic trust anchor store update mechanisms, with different versions of Windows providing somewhat different capabilities. When these features are enabled, a trust anchor may be automatically installed with no visual cue provided to the operator, for example, when a certificate file subordinate to that trust anchor is simply inspected using the Windows certificate viewer a corresponding trust anchor may be downloaded and installed. Trust anchors installed automatically do not necessarily have all purposes enabled.

Trust anchor stores are maintained in the system registry. Trust anchors are imported and exported as certificates. The certificates are stored in the registry along with property information. When trust anchors are exported, the user-configured constraints are not conveyed along with the exported certificates.

2.1.2 Firefox

Firefox does not use Microsoft Windows trust anchor stores. Trust anchors are added to the Firefox trust anchor store using the Certificate Manager dialog shown below in Figure 3. This dialog is accessed by invoking the Tools->Options menu and selection the Encryption tab from the Advanced options.

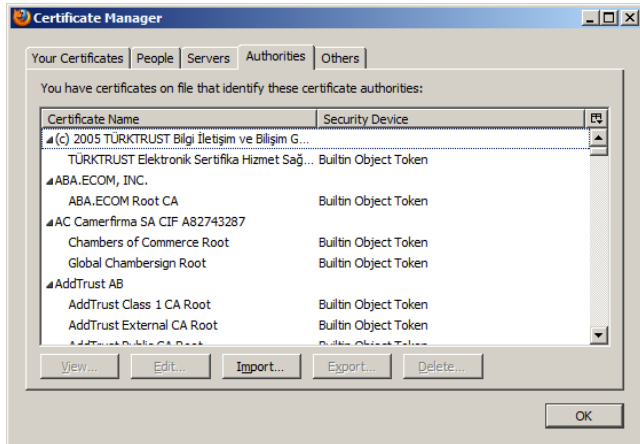


Figure 3 Firefox trust anchor store

Trust anchor constraints can be configured by clicking the Edit button and selecting the desired properties in a dialog like the one shown below, which allows three properties to be enabled. As with Microsoft Windows, the properties are similar to values that are typically expressed via an extended key usage extension.

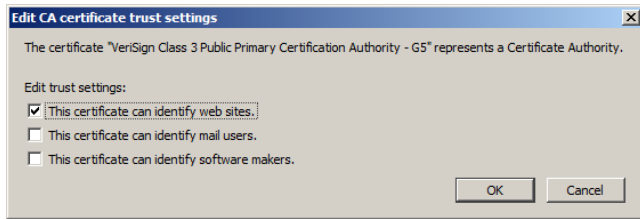


Figure 4 Firefox trust anchor constraints

Firefox trust anchors are maintained in a database that resides in the profile of a Firefox user. Trust anchors are imported and exported as certificates.

2.1.3 Mac OS X

Mac OS X maintains trust anchors in the key chain. Trust anchors are added to the trust anchor by invoking the Keychain Access application, as shown below.

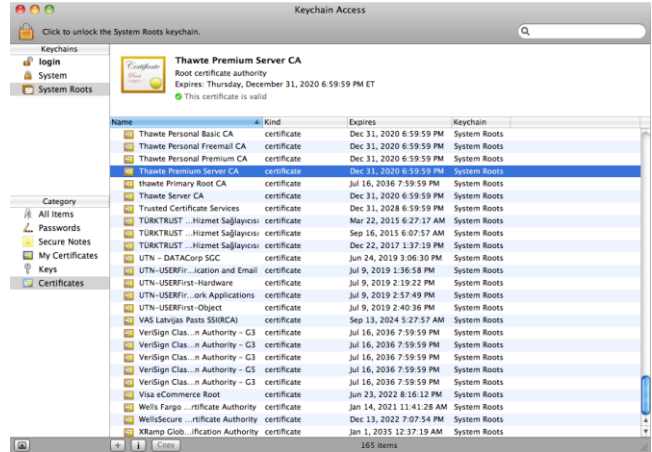


Figure 5 Mac OS X version 10.6 trust anchor store

Trust anchor information, including usage constraints, can be viewed by right-clicking a trust anchor in the Keychain Access application and choosing Get Info. Nine properties are available. As with Microsoft Windows and Firefox, the properties are very similar to values typically expressed via an extended key usage extension.

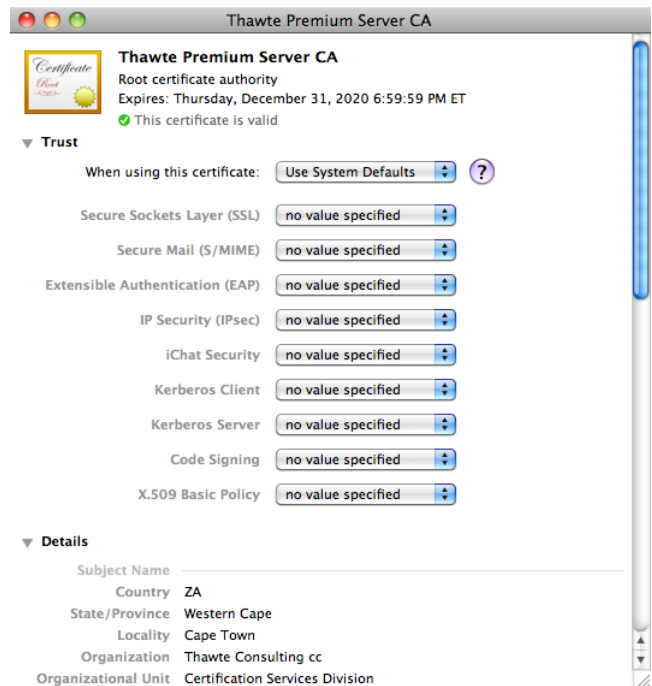


Figure 6 Mac OS X version 10.6 trust anchor constraints

As with Microsoft Windows and Mozilla trust anchor stores, trust anchors are exported as files containing X.509 certificates, and no user-specified constraints are conveyed along with these certificate files.

2.2 Primary problems with current mechanisms

This paper does not aim to catalog problems with existing trust anchor management mechanisms. However, this section discusses

some problems in the areas of trust anchor store management and trust anchor constraints enforcement.

2.2.1 Trust anchor store management

Management of trust anchor stores requires usage of proprietary tools. Where necessary, system administrators must take care to synchronize the contents of multiple trust anchor stores. This requires configuration of trust anchor constraints as well as ensuring trust anchors are installed in (or removed from) the necessary trust anchor stores.

Maintenance of trust anchor store contents is complicated by the fact that software updates frequently adjust trust anchor store contents (sometimes undoing changes made by the system administrator). Automatic trust anchor update mechanisms can create de facto trust anchor stores that contain more trust anchors than are visible to administrators using the available tools.

Trust anchors do not offer any integrity protection or “in-band” security mechanisms. Confirmation that the correct trust anchor is being installed typically requires manual checks.

2.2.2 Constraint representation

As shown in Section 2.1, different trust anchor stores enable the usage of different, non-standard trust anchor constraints. These constraints are stored using a proprietary format. When trust anchors are exported from the trust anchor store the constraint information is lost.

The certification path validation algorithm described in RFC 5280 [1] only makes use of the public key and name of a trust anchor. Implementations are free to perform processing beyond that required by RFC 5280 [1], such as to impose name constraints or certificate policy requirements on a trust anchor. However, there is no standardized process for doing so. This lack of standardization has resulted in inconsistent means of specifying constraints and poor interoperability. Complicating matters is the fact that trust anchors are almost always represented as certificates. Though the signature on the trust anchor’s certificate provides little security value, it interferes the editing of certificate contents.

2.2.3 Constraint enforcement

Enterprise PKI operators use cross-certificates to establish trust between enterprises and employ a variety of constraints, i.e., extensions, to limit the degree of trust accorded to the cross-certified PKI. However, cross-certificates are not always a viable option. In some cases, however, a trust relationship may only be appropriate for a small subset of subscribers to an Enterprise PKI. In these cases, directly trusting a trust anchor is an alternative. Unfortunately, existing trust anchor constraint mechanisms do not provide a set of constraint options comparable to those available when using a cross-certificate, making direct trust difficult to use.

For an example of problems caused by lack of trust anchor constraints, consider the community surrounding the Federal

Bridge CA (FBCA). Each CA that has issued a cross-certificate to the FBCA creates a large number of potential certification paths that traverse that cross-certificate. Some enterprises, such as the Department of Defense, have adopted an approach to cross-certifying with the FBCA that allows application owners to opt out of the cross-certification by recognizing alternative trust anchors that are not connected to the FBCA. A problem arises when entities who have “opted out” need to establish a trust relationship with another CA that is cross-certified with the FBCA. Simply recognizing the CA as a trust anchor will establish the trust relationship but causes the entire FBCA community to be recognized as well. This could be avoided if it were possible to constrain a trust anchor using similar mechanisms as those used in cross-certificates.

3. Next Generation Specifications

The Internet Engineering Task Force (IETF) is presently working on several specifications related to trust anchor management and usage, including: Trust Anchor Management Protocol (TAMP) [4], Trust Anchor Format (TAF) [3], CMS Content Constraints (CCC) [2], Using Trust Anchor Constraints during Certification Path Processing (UTAC) [5]. These specifications provide complementary features, but subsets of features can be implemented where the full feature set is not required.

The following subsection briefly introduce each of these specifications, which were used in the implementation described in Section 4.

3.1 Trust Anchor Format

TAF [3] provides syntax for representing trust anchors. The primary structure is TrustAnchorChoice:

```
TrustAnchorChoice ::= CHOICE {  
    cert Certificate,  
    tbsCert [1] EXPLICIT TBSCertificate,  
    taInfo [2] EXPLICIT TrustAnchorInfo  
}
```

This structure provides support for existing trust anchors represented as certificates and provides two mechanisms that allow relying parties to customize the definition of a trust anchor: TBSCertificate and TrustAnchorInfo. Using the TBSCertificate option, the signature is simply removed from a Certificate structure allowing the contents to be edited. Using TrustAnchorInfo, a Certificate can be wrapped, with additional or alternative constraints defined in the wrapper or a name and public key can be used with or without additional information.

3.2 Trust Anchor Management Protocol

TAMP [4] defines eleven message formats and a set of processing rules that can be used to manage trust anchor store contents. Each of these message formats, or content types, can be encapsulated using a CMS SignedData structure to provide source authentication and message integrity. The eleven messages consist of five request/response pairs and a generic error message:

- TAMPStatusRequest
- TAMPStatusResponse
- TAMPUpdate
- TAMPUpdateConfirm

- TAMPApexUpdate
- TAMPApexUpdateConfirm
- TAMPCommunityUpdate
- TAMPCommunityUpdateConfirm
- SequenceNumberAdjust
- SequenceNumberAdjustConfirm
- TAMPErrror

3.2.1 Reviewing TA store contents

TAMPStatusResponse messages provide a means of representing trust anchor store contents. As with most TAMP response/confirm messages, the message can be either verbose or terse. A verbose TAMPStatusResponse message provides a comprehensive set of information regarding a trust anchor store, including a list of all trust anchors, an indication of which TA is the apex trust anchor (if any) and information on TAMP sequence numbers and TAMP communities. A terse TAMPStatusResponse provides only trust anchor key ids along with communities of which the store is a member. A TAMPStatusRequest simply asks a trust anchor store to provide its contents in the requested message format, i.e., verbose or terse. Use of TAMPStatusRequest and TAMPStatusResponse can reduce reliance on proprietary tools for TA store management and simplify comparison of TA store contents.

3.2.2 Editing TA store contents

TAMPUpdate messages allow new trust anchors to be added to a trust anchor store, existing trust anchors to be changed or existing trust anchors to be removed. Each TAMPUpdate message contains a set of one or more commands (i.e., add, change, remove). Since TAMPUpdate messages are signed, in-band integrity and source authentication checking is enabled.

3.2.2.1 Subordination rules

TAMP defines a strict set of subordination rules that apply when a TAMPUpdate message is processed. These rules allow limits to be placed on TA store managers. These rules could be used to place constraints on automated updates, such as to ensure an undesirable trust anchor is not restored after it has been removed by a local management action, or to ensure that a trust anchor rekey operation does not exceed locally-imposed constraints on the old key.

3.2.3 Replacing the Apex TA

TAMP [4] introduces the concept of the Apex TA, which is defined as being the single trust anchor within a trust anchor store that is superior to all other trust anchors. This concept is primarily used as a disaster recovery technique. Essentially, a trust anchor store is created with a single Apex TA in place. Authority over various management operations is then delegated to other trust anchors that are added to the trust anchor store or to certificate holders. Management operations are conducted by the delegates with the Apex TA private key maintained in secure storage. As an extra safeguard, a contingency public key can be included in the definition of the Apex TA. The contingency public key corresponds to a private key that is intended to be used once to replace the Apex TA in the event of loss or compromise of the operational Apex TA private key.

3.2.4 Managing TAMP community membership

TAMP messages can be created such that all TA stores that recognize the TA store manager will accept the message, a group of TA stores will accept the message or a specific TA store will accept the message. Community identifiers are one means for addressing a group of trust anchor stores. TAMP-enabled trust anchor stores should have the ability to store a list of community identifiers. TA store managers can use these identifiers to create arbitrary groups of trust anchor stores for future management purposes.

TAMPCommunityUpdate messages are used to add or remove community identifiers from a trust anchor store. TAMPCommunityUpdateConfirm is used to report the results of processing a TAMPCommunityUpdate message.

3.2.5 Managing TAMP sequence numbers

TAMP uses sequence numbers to detect attempts to process old TAMP messages. Each TAMP-enabled trust anchor store maintains a sequence number for each trust anchor authorized for TAMP (and may maintain a sequence number for certificate holders who have been authorized for TAMP). A SequenceNumberAdjust message can be used to convey the current sequence number to a trust anchor store to reduce the likelihood of replay. A SequenceNumberConfirm message is used to indicate the results of processing the SequenceNumberAdjust message.

3.3 CMS Content Constraints

A basic problem for any trust anchor management protocol is authorization of management operations. Certification authorities are authorized to issue cross-certificates using constraints expressed as certificate extensions, e.g., basicConstraints, certificatePolicies, etc. CCC [2] defines an authorization mechanism that can be used with TAMP.

CCC is a generic mechanism for authorizing public key certificate holders to originate specific types of information protected using the Cryptographic Message Syntax (CMS). A set of content types is expressed in the CCC extension. When a CMS-protected message is processed, the originator is authenticated and the CCC extension associated with the originator is inspected to ensure the given content type is permitted.

For TAMP, this mechanism can be used to authorize some entities to manage trust anchor stores and others to review the contents of trust anchor stores while leaving other entities with no privileges at all. To authorize an entity to manage trust anchor stores, include, in either the entity's certificate or trust anchor, a CCC extension with the TAMPUpdate, CommunityUpdate, SequenceNumberAdjust and TAMPStatusQuery content types permitted. To authorize an entity to review the contents of trust anchor stores, include a CCC extension in the entity's trust anchor or certificate with the TAMPStatusQuery content type permitted.

3.4 Using Trust Anchor Constraints during Certification Path Processing

UTAC [5] augments the certification path processing algorithm specified in RFC 5280 [1] by describing how to use constraints contained in a trust anchor during certification path processing. Essentially, the constraints contained in a trust anchor are intersected with those provided by a user. The results of this intersection are used as the inputs to the RFC 5280 [1] certification path validation algorithm. This allows a trust anchor store manager (i.e., an enterprise) to establish a minimum set of restrictions on the usage of a trust anchor without removing the ability of an application (i.e., a user) to provide inputs to the path validation algorithm.

UTAC [5] describes rules for using constraints in a TrustAnchorInfo wrapper relative to constraints resident in a certificate that is wrapped, i.e., the wrapper takes precedence. UTAC processing can be integrated directly into an RFC 5280 path validation implementation or as pre or post processing.

4. Integrating Trust Anchor Management with CAPI

The goal of the implementation effort described in this paper was to enable the usage of emerging trust anchor management specifications with commonly deployed commercial off-the-shelf (COTS) products which have been public key-enabled using Microsoft Crypto API (CAPI). This integration aims to enforce constraints associated with a trust anchor. To achieve this, the software must be able to influence the outcome of a certification path validation operation performed by CAPI.

Since there is no publicly documented set of APIs intended for this purpose, existing APIs intended for other purposes were evaluated to determine suitability for integration of trust anchor management functionality. The following interfaces were analyzed: revocation status provider, validation policy provider and certificate store provider.

4.1 Revocation Status Provider

The initial approach that was considered was to use the revocation status provider interface. Revocation status providers are typically used to provide support for Online Certificate Status Protocol (OCSP). A revocation status provider is a dynamic link library (DLL) that implements the CertVerifyRevocation API. The provider is registered with the operating system. The registration information consists of the full path and filename of the revocation status provider and is stored in a registry key containing a list of string values. The list of providers can be ordered according to system administrator preference. Providers are invoked in turn until a one is found that can provide revocation status information for the certificate in question.

When an application validates a certification path, the provider is loaded by CAPI and invoked once for the end entity certificate and each intermediate CA certificate contained in a certification path validated by CertGetCertificateChain or WinVerifyTrust.

The provider can cause a path validation operation to fail by indicating the given certificate is revoked.

This approach was not implemented for two reasons. First, the interface is invoked for each certificate in a path, not for an entire certification path. This means the provider would need to maintain state across multiple invocations in order to get a view of the entire path. Second, while this could effectively cause a certification path that violates trust anchor constraints to fail, the error indicated by the provider creates a misimpression that a certificate is revoked. This kind of misreported failure leads to a poor user experience in the desktop applications that are targeted in this effort.

4.2 Validation Policy Provider

Next, the validation policy provider interface was explored. This interface is not as comprehensively documented and less widely used than the revocation status provider interface. Like the revocation status provider interface, a validation policy provider is registered with the operating system and loaded by CAPI during certification path processing. Unlike the revocation status provider interface, providers do not failover from one to another. Providers can be registered for a specific validation policy. However, the processing performed by default policy providers is not documented and replacing the default providers is not recommended. No way could be found to invoke the default providers from a third party provider.

We implemented policy providers for several of the default policies but abandoned the effort due to inconsistent invocation of the installed replacement policy provider. For example, within Microsoft Outlook, the replacement policy provider was invoked when no certification path was found for a message signer but not when a certification path was found.

4.3 Certificate Store Provider

While performing the analysis of the validation policy provider API, we used code interception to inspect and log parameter values. After discarding the revocation status provider and validation policy provider efforts, we focused on finding a means of using code interception as the basis for performing the integration. This required identifying opportunities where code could be loaded prior to the CertGetCertificateChain API and unloaded afterwards, enabling the CertGetCertificateChain to be intercepted. For most applications, the certificate store API provides such an opportunity.

We implemented a certificate store provider that is registered with the operating system as a CA store provider in the HKEY_LOCAL_MACHINE registry hive. When the certificate store is loaded, hooks are created for the CertGetCertificateChain API. No certificate store functionality is actually provided.

To limit the scope of the provider, configuration information can be saved on a per-application basis. When an application that does not require the trust anchor management services implemented by the provider loads it, no hooks are set. The

certificate store provider is loaded into memory but performs no code interception.

A side benefit of this integration approach is the ability to fully replace CAPI certification path processing instead of simply enforcing trust anchor constraints following discovery of a certification path. This enables the usage of the Server-based Certificate Validation Protocol (SCVP) or alternative local certification path processing engines for both path discovery and validation. Though the software described below supports this option, it is not discussed further in this paper. Nor are issues associated exclusively with the provision of SCVP support.

As noted above, integration via the certificate store API proved workable for most applications that were tested but not all. For Internet Explorer, it was necessary to build a browser add-on that causes the certificate store to be loaded before the browser can be used to access SSL/TLS-protected websites. The browser add-on simply forces CertOpenStore to be called by validating a path to a hard-coded trust anchor, which was selected from the list of required trust anchors defined in Microsoft knowledge base article number 293781.

5. CAPI Trust Anchor Guard (CAPI TAG)

CAPI Trust Anchor Guard (CAPI TAG) is a set of software tools that enable management of a local or remote trust anchor store using TAMP and enforcement of trust anchor-based constraints for applications that use CAPI for certification path processing.

5.1 Overview

CAPI TAG consists of eight primary components: PKIFTAM, CAPI TAG Store Creator, Store Manager, mod_tam, Process TAMP Message, CAPI TAG, CAPI TAG Config and CAPI TAG Customization Wizard.

5.1.1 PKIFTAM

PKIFTAM.dll provides basic encoding and decoding functionality for structures defined in TAF [3], TAMP [4] and CCC [2]. Additionally, it provides classes that can be integrated with the PKIF library (www.pkiframework.com) to enforce TA constraints using a TA store managed with TAMP.

5.1.2 CAPI TAG Store Creator

CapiTagStoreCreator.exe is used to initialize a CAPI TAG trust anchor store. A trust anchor store can be created using trust anchors from a CAPI trust anchor store or a file folder.

5.1.3 Store Manager

StoreManager.exe is the primary trust anchor management tool. It can be used to manage local trust anchor stores, remote trust anchor stores accessed via HTTP or remote trust anchor stores via a file containing a TAMPStatusResponse message generated by the target trust anchor store. The user interface in Store Manager is mostly driven by TAMP messages, and all operations are possible regardless of access method, provided the operator

possesses an authorized signing key. The primary interface to manage trust anchors using Store Manager is shown below.

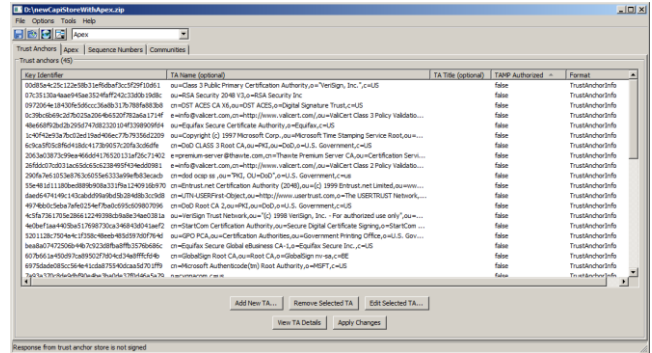


Figure 7 Store Manager trust anchor list

Using Store Manager, trust anchors can be added to a TA store, removed from a TA store or edited. When a trust anchor is added, its format can be changed from certificate to TBSCertificate or TrustAnchorInfo, enabling the expression or alteration of constraints.

Trust anchor constraints are edited using dialogs provided with the PKIF library. These allow the expression of constraints that align with the standard path validation algorithm inputs as defined in RFC 5280 [1]. The constraints editing dialog is shown below.

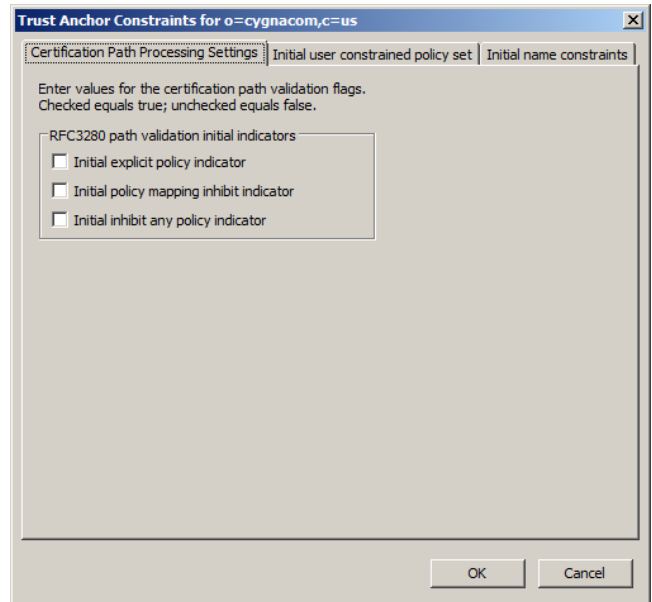


Figure 8 Editing trust anchor constraints in Store Manager

5.1.4 mod_tam

mod_tam is an Apache module that serves either or both of the following purposes:

- Routes TAMP messages received via a particular URI to a TA store file for processing
- Periodically check specified URIs for TAMP messages, which are downloaded and presented to a TA store file for processing.

This enables the suite to support either push or pull for TA management. mod_tam is accompanied by an optional system tray notification applet that allows the user to see desktop alerts as TAMP messages are processed.

5.1.5 Process TAMP Message

ProcessTampMessage.exe allows a file containing a TAMP message to be presented to a CAPI TAG trust anchor store for processing. The store can be addressed either as a local file or using an HTTP URI. Unlike Store Manager, the operator of Process TAMP Message need not have any TAMP privileges (or even possess a private key).

5.1.6 CAPI TAG

CapiTag.dll integrates with Microsoft Windows operating systems to provide trust anchor constraints enforcement or alternative certification path processing.

5.1.7 CAPI TAG Config

CapiTagConfig.exe is the primary means for configuring CapiTag.dll for use. It enables the configuration of default settings and application-specific settings. All configuration information is stored in the system registry.

5.1.8 CAPI TAG Customization Wizard

CapiTagCustomizationWizard.exe is used to create transform files (.mst) that can be used to customize the CapiTag.msi installation package for use in a particular environment. The wizard allows customization of the following aspects of a CAPI TAG deployment:

- Inclusion of one or more CAPI TAG trust anchor stores
- Customization of Store Manager PKI settings (i.e., used when validating TAMP messages generated by a CAPI TAG TA store)
- Customization of CAPI TAG trust anchor store PKI settings (i.e., used when validating TAMP messages generated by Store Manager)
- Customization of CAPI TAG PKI settings (i.e., used when enforcing TA constraints or to configure alternative certification path processing)
- Customization of CAPI TAG settings (i.e., default or per-application settings)
- Specification of a customized mod_tam configuration file

5.2 Trust Anchor Management

Using CAPI TAG, several trust anchor management models are possible. As shown in Figure 9, the management models considered here are: local management, online remote management, indirect remote management and remote pull. The terms local and remote refer to the relative positions of the trust anchor store and the trust anchor manager's private key. For local management scenarios, the TA store and TA store manager's private key are collocated¹. For remote management scenarios, the TA store and TA store manager's private key need not be collocated.

Given that CAPI TAG trust anchor stores are files, the contents could be prepared in one location and distributed using means like group policy. With minor additions to the current specification suite, additional models including usage of a subjectInformationAccess-based pointer or trust anchor store-initiated client/server exchange are possible.

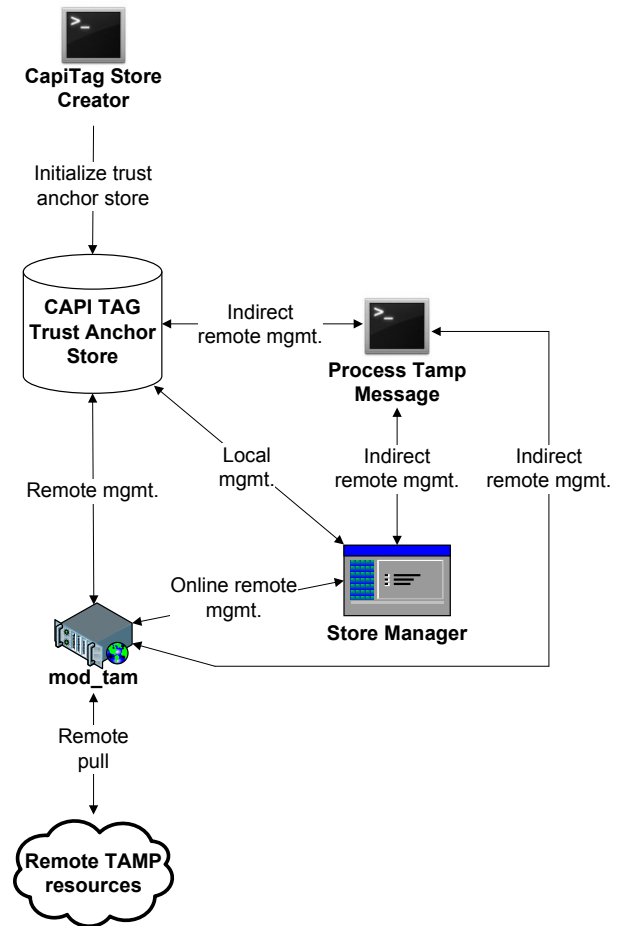


Figure 9 Management models

¹ This taxonomy is quite loose. Files accessed over a local area network are considered “local” despite the fact that the TA store resides on a different physical machine. Similarly, files accessed over HTTP to a local mod_tam service are considered “remote”.

5.2.1 Local management

Using the Store Manager application, a CAPI TAG trust anchor store file can be opened and queried using a TAMPStatusQuery message. The Store Manager operator's private key must be available and the target trust anchor store must recognize the operator as authorized to originate TAMPStatusQuery messages (no other permissions are required to simply review the contents of a TA store).

If the operator is authorized to edit TA store contents, changes can be made and saved using Store Manager.

5.2.2 Online remote management

Using the Store Manager application, a CAPI TAG trust anchor store can be managed via HTTP by entering the URI corresponding to the desired trust anchor store. This will establish a connection to a mod_tam service, which will route TAMP messages to/from trust anchor stores collocated with the mod_tam service per the httpd.conf file.

As with local management, the operator may be authorized to edit the TA store or simply to review the TA store contents.

5.2.3 Indirect remote management

TA stores can be managed remotely through exchange of files containing TAMP messages. An entity with at least TAMPStatusQuery privileges can generate a TAMPStatusResponse message using Store Manager. The file containing the response can be provided to another entity with full TAMP privileges, who can then open the file using Store Manager and generate one or more TAMP messages to edit the TA store. These messages can be returned to the requesting entity for processing using the Process TAMP Message utility. To ensure security, the TA store should sign the TAMPStatusResponse.

5.2.4 Remote pull

A TA store manager can prepare TAMP messages using Store Manager for distribution via HTTP. The mod_tam service can be configured to periodically retrieve TAMP messages for zero or more URIs for processing by the indicated trust anchor store associated with the mod_tam instance.

In CAPI TAG, automated remote pull is not available without the mod_tam service. TAMP messages can be manually collected and processed using either ProcessTAMPMessage or the process externally generated TAMP message feature of Store Manager.

5.3 Trust Anchor Constraints Enforcement

CAPI TAG can be configured to enforce trust anchor constraints on a per-machine, per-user or per-application basis. When an application loads CAPI TAG, the most specific available configuration is used. The order of preference is as follows:

- Current user – application
- Local machine – application
- Current user – default

- Local machine – default

This allows a high degree of configurability for trust anchor stores and application PKI settings. Some applications can be configured to enforce trust anchor constraints, others can be configured to use an SCVP responder (or alternative local certification path processing implementation) and other applications can be configured to use native processing without TA constraints enforcement. This degree of configurability makes it easy to enforce constraints for key applications without impacting any legacy incompatible applications that need to run on the same system.

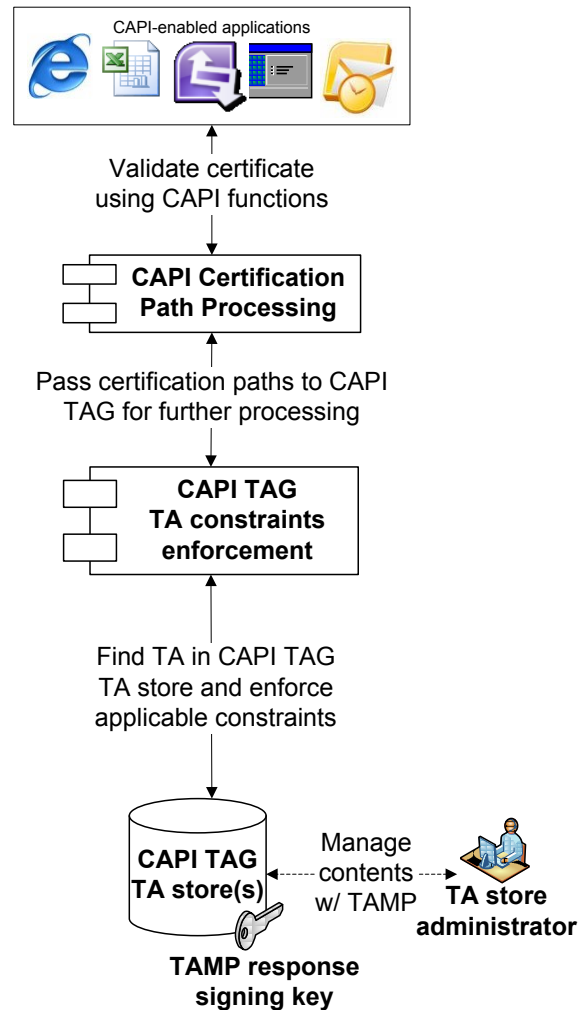


Figure 10 TA constraints enforcement with CAPI TAG

Since CAPI TAG uses a trust anchor store that is separate from CAPI trust anchor stores, the CAPI TAG trust anchor store manager's actions are not affected by changes made to the CAPI trust anchor store via automated trust anchor store updates or software upgrades. CAPI TAG can be configured to accept trust anchors from CAPI when a path is validated to a trust anchor not present in the CAPI TAG trust anchor store and can be configured to write trust anchors to a file folder, enabling the trust anchor store manager to adjust the contents of the CAPI TAG trust

anchor store as necessary. This feature reduces the difficulty of determining which trust anchors must be present and trusted to ensure that an application continues to function as users expect.

CAPI TAG can also be configured to not act for certain types of operations. For example, CAPI TAG can be configured to use only native functionality when a certificate is validated in support of a CAPI trust root list validation operation.

6. Summary

CAPI TAG demonstrates the effectiveness of the emerging IETF trust anchor management specifications in a typical, commercial software environment. CAPI TAG is intended to generate interest and discussion in trust anchor management and usage practices that ensure relying party interests can be satisfied. This section describes some challenges encountered while developing the CAPI TAG products and identifies some areas where additional standardization is potentially required.

6.1 Implementation experience

A primary challenge encountered during the development of the software was the lack of a proper interface for integrating enhanced trust anchor management capabilities and enforcement of trust anchor constraints. Not surprisingly, once an approach was identified it was also proved suitable for implementing an SCVP client. Most of the problems associated with the selected integration mechanism could easily be addressed if a means of utilizing alternative certification path processing implementations similar to that used for installing alternative revocation status providers were available.

Integration of non-certificate formats into a trust anchor store posed another challenge. This was solved by using a CAPI TAG-specific trust anchor store file format. Several challenges prevented the usage of existing mechanisms. The interfaces to existing trust anchor stores accept (usually self-signed) certificates. Trust anchor management messages were the desired format to support in-band integrity checks, authorization, subordination checks, etc. While it may have been possible to have overloaded the `CertAddEncodedCertificateToStore` to handle TAMP messages, this was not explored. For these reasons, TA store management was implemented as wholly independent of CAPI.

Read/write access to the trust anchor store file is managed by the operating system. CAPI TAG trust anchor store usage only requires read access. Write access can be limited to the `mod_tam` service, if desired. By default, though, system administrators have write access to CAPI TAG trust anchor store files. Authorization to manage trust anchor store contents via a TAMP interface is enforced using CCC.

Trust anchor constraints enforcement was integrated with an existing public key enablement library (PKIF). Integration of support for alternative formats [3] required a number of changes to the library. These were addressed primarily through the use of abstract interfaces that captured the common elements of the various formats, i.e., all featured a subject name, a public key and extension values.

The SCVP-client mode of operation in CAPI TAG required the availability of certificates in order to use existing structures that could not be changed. For CAPI TAG purposes, trust anchors are always represented as either a certificate or a `TrustAnchorInfo` containing a certificate. It may have been possible to recast trust anchors stored as `TBSCertificate` or `TrustAnchorInfo` objects as `Certificates` with bogus signatures, but this was not explored.

Integration of trust anchor constraints enforcement [5] with the PKIF library was straightforward. Initially support was integrated as wrapper code that resided in an application, but this was moved into the library itself and exposed as an optional feature of the path validation implementation. Constraints enforcement [5] can be implemented independent of other trust anchor management specifications [2][3][4] using extensions expressed in self-signed certificates. This would be of limited utility at present given the fact that most self-signed certificates do not include constraints of any sort.

At a high level, the implementation of support for the trust anchor management specifications and integration of that support into existing products consisted of the following activities:

- Define trust anchor store format
- Define and implement trust anchor store interface and access control mechanisms
- Identify code that uses trust anchors and make adjustments to accommodate new formats, where necessary
- Implement trust anchor constraints enforcement as pre/post processing of path validation or integrate with path validation code

Following the implementation of support for trust anchor management and trust anchor constraints enforcement, deployment of the capabilities consisted of the following activities:

- Identify the applications of interest (i.e., web browsers, email clients, etc.)
- Identify the trust anchors required by these applications
- Identify entities authorized to manage trust anchor stores
- Initialize trust anchor stores to include desired trust anchors (including constraints) and trust anchor store managers
- Distribute trust anchor stores and enable trust anchor constraint enforcement capabilities
- Manage trust anchor stores using appropriate local, remote, direct or indirect means

6.2 Potential additional standardization needs

Most existing trust anchor constraints mechanisms provide a capability similar to the extended key usage extension. Unfortunately, extended key usage values included in a trust anchor are not processed during certification path validation [1]. Defining an extension and an augmentation of the standard path validation algorithm would be simple and straightforward and

potentially valuable in terms of promoting interoperability. However, the utility of this extension is not entirely clear given that most enterprises do not operate certificate authorities, let alone root certification authorities, on a per extended key usage basis.

The usage of the existing name constraints extensions in trust anchors is effective in enterprise environments where naming conventions are rigorously controlled and are generally hierarchically related. The name constraints mechanism is less suited to internet use, where distinguished names vary greatly within a single certification path and server names are often conveyed as a terminal relative distinguished name (RDN) value. Addressing this issue may be more easily accomplished by refining naming practices to enable the usage of existing name constraints mechanisms than defining alternative constraint mechanisms.

Another name constraints-related issue is the observation that to effectively use name constraints, most or all trust anchors in a given trust anchor store must have an associated name constraint value. To ensure that a particular namespace can only be issued by a given trust anchor all other trust anchors must be defined to either have an alternative permitted namespace or to exclude the namespace of interest.

Though no in-depth investigation of the utility of trust anchor management tools to counter phishing attacks was conducted, it is possible that better use of existing constraints or definition and

adoption of additional constraints could provide useful countermeasures.

7. REFERENCES

- [1] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
- [2] Housley, R., Wallace, C., and S. Ashmore, "Cryptographic Message Syntax (CMS) Content Constraints Extension", in progress.
- [3] Housley, R., Wallace, C., and S. Ashmore, "Trust Anchor Format", in progress.
- [4] Housley, R., Wallace, C., and S. Ashmore, "Trust Anchor Management Protocol (TAMP)", in progress.
- [5] Wallace, C. and S. Ashmore, "Using Trust Anchor Constraints During Certification Path Processing", in progress.
- [6] Wallace, C. and R. Reddy, "Trust Anchor Management Requirements", in progress.