Secure Management of Objects in Remote Containers Shared by Multiple and Independent Issuers - II

Background to the Invention

- Issuing entities issue objects to object consumers
- An object consumer may be a consumer of objects from multiple issuers while keeping received objects in a single secure container
- Issuers are supposed to be the actual owner (manager) of the objects they issue, while object consumers typically only are "users" of such objects
- Issuers do in most cases not have direct access to the object consumers' secure containers, but rather depend on remote (on-line) object issuance procedures.

Put into real-world terms, issuing entities could be financial institutions, while object consumers could be individuals equipped with virtual credit-cards (objects) stored and executing inside of TPMs (Trusted Platform Modules).

The object consumer container could advantageously be an integral part of a mobile computer or mobile phone, giving a multitude of object issuance and usage possibilities.

Problem Description

If an issuer wishes to *replace*, *modify*, *clone*, or *delete* a *previously issued object* in a remote container *shared* by *multiple and independent issuers*, there are a number of requirements needing attention for making this viable:

1. Aiding the object consumer in selecting the proper target object
2. Limiting an issuer's ability manipulating objects it does not own (is not the issuer of)

References

This is a substantially revised version of an earlier invention with the same subject published as:

http://ip.com/IPCOM/000179109

The primary enhancement is that Object Lookup and Object Management have been separated and that symmetric keys are used for the latter, making the concept more *robust* as well as *compliant with resource-constrained secure containers*.

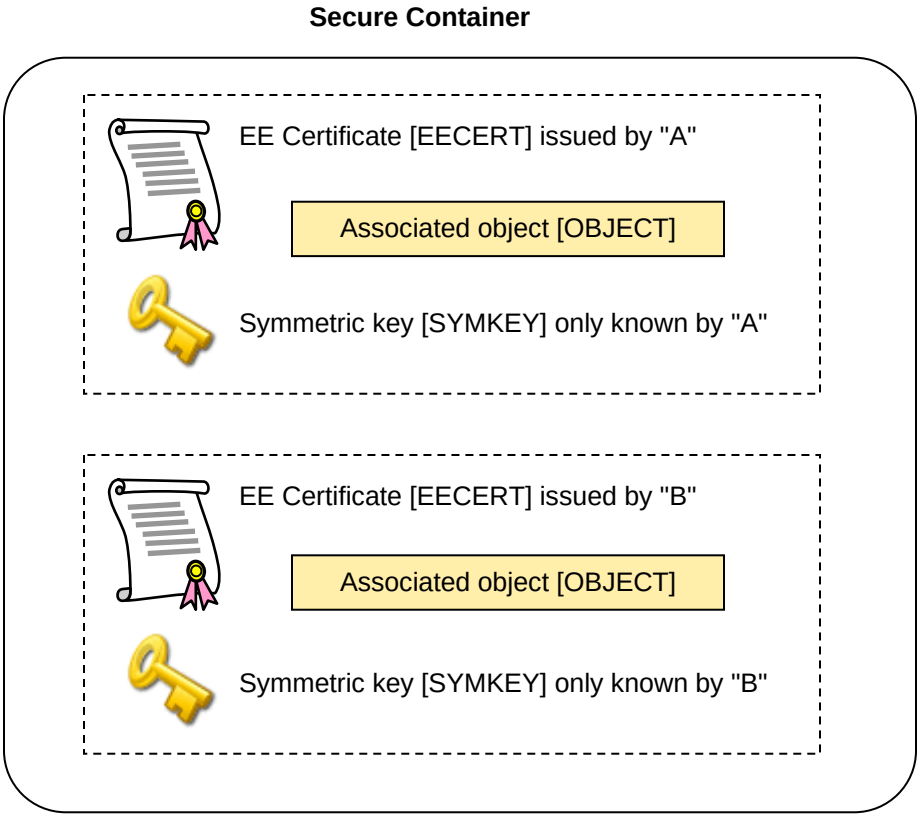
Proposed Solution

Notation: Items within brackets [] denote elements in the pictures featured in this specification.

Each issued object [OBJECT] is augmented by an X.509 EE (end-entity) certificate [EECERT] serving as a GUID (Globally Unique ID) with the added capability that it can be securely bound to a specific CA (issuer). In addition, each issued object [OBJECT] is also linked to a symmetric key [SYMKEY] that is only known by the issuer and the secure container.

> *Note: How [OBJECT], [EECERT] and [SYMKEY] originally are provisioned to secure containers is covered by other specifications. This specification exclusively deals with post-provisioning updates.*

Below is an example of a pair of objects issued to an object consumer by issuer "A" and "B" respectively:



Now assume that issuer "A" wants to manage the object above which it has already issued.

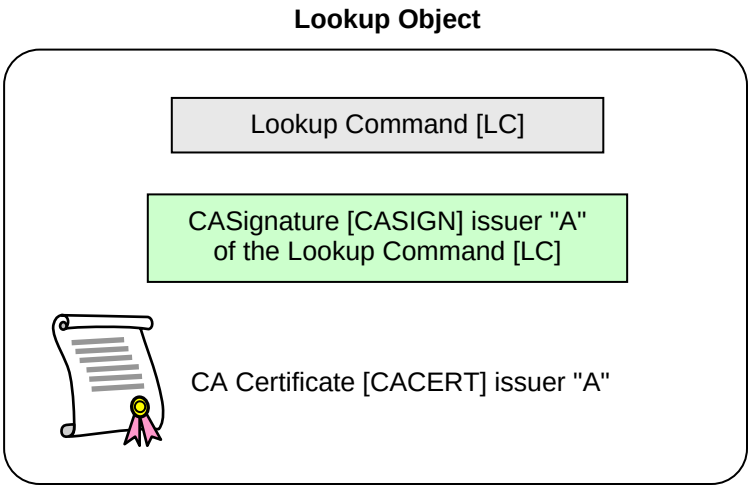
Continued on the next page…

Optional Phase – Lookup the State of the Target Object

Since an object *may* to some extent be manageable by the object consumer (user) who typically could delete an object, this proposal contains an *optional* phase where the issuer in a secure and privacy-conscious way can “lookup” objects it (and only it) has issued.

The value of doing a lookup before performing management operations is that remote diagnostics of the actual situation increase robustness of the update process and may even lead to a controlled abort of the process where the user is informed that there is something wrong and what to do about it.

To perform a lookup the issuer applies the CA private key (not shown here) and the CA certificate [CACERT] it used when issuing the object(s) end-entity certificate [EECERT], but this time for signing [CASIGN] a Lookup Command [LC] according to the following:



The completed lookup object is subsequently sent to the object consumer for execution.

Due to the fact that a Lookup Command [LC] is uniquely bound to an issuer by the signature [CASIGN] and CA certificate [CACERT], the object consumer (software) system can automatically and securely restrict lookup operations to only apply to objects with matching end-entity certificates [EECERT]. I.e., in addition to a valid CA signature [CASIGN], [CACERT] and the [EECERT] of the target [OBJECT] must form a valid two-element certificate-path in order to enable lookups.

Note that there is no need for object consumers "trusting" an issuer by having the issuer's root certificate installed. This is compliant with, for example, credit-cards which do not have to be verified by consumers, only by relying parties (merchants).

The Lookup Command [LC] may be parameterized to not list all objects associated with the particular issuer. Since X.509 end-entity certificates [EECERT] support a number of standardized attributes, a Lookup Command [LC] may specify object [OBJECT] selection attribute patterns like:

```
Subject = "CN=John Doe,O=example.com,C=US"
Email = "*@example.com"
```

The result of a Lookup Command [LC] is sent back to the issuer as a list of SHA1 “fingerprints” of the matching end-entity certificates [EECERT].

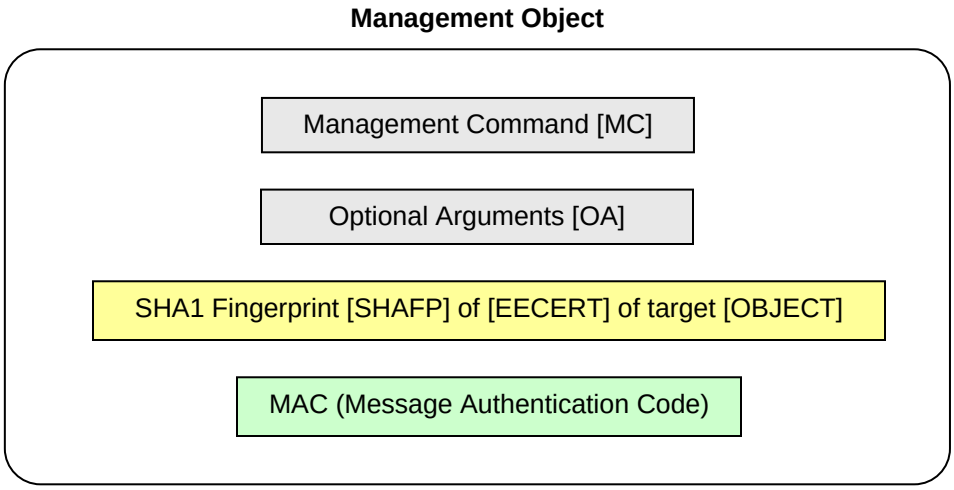
Since a lookup is passive (does not modify data), does not depend on secret keys in the secure container, and is rather processor-intensive, lookups may with advantage be performed by the middleware layer which typically is more powerful than a secure container processor.

A further advantage of the lookup procedure is that it enables updates without necessarily authenticating the object consumer since the retrieved object fingerprints effectively constitute of indirect authentication data as well.

Executing Management Commands

In order to execute management commands, the issuer must know the SHA1 fingerprints [SHAFP] of the end-entity certificates [EECERT] of the objects [OBJECT] it wants to manage. This may either be known a priori or be gathered through the previously described Lookup Command [LC].

To perform a management operation the issuer creates a command according to the following:



Where the MAC is calculated by the formula

```
HMAC_SHA1 (SYMKEY, MC || OA || SHAFP)
```

which presumes that the issuer keeps track of all [SYMKEY] in order to be able to perform post-provisioning updates.

The completed management object is subsequently sent to the object consumer for execution.

Due to the fact that a Management Command [MC] is uniquely bound to the [EECERT] by [SYMKEY], the object consumer can validate the command object and if it is valid perform the operation. Since the scheme uses simple symmetric key operations, the execution may be performed *inside* the secure container which means that [SYMKEY] does never have to be exposed to the potentially untrusted middleware.

Implementation

The described scheme has been used for creating a universal "key-ring" supported by a universal key issuance and management protocol, where the end-entity certificates [EECERT] function as universal object-IDs for PKI keys, symmetric keys, PIN and PUK policy objects, key-attributes, logotypes, crypto-algorithm code, arbitrary extension data, as well as for meta-data schemes like Microsoft Information Cards.

Version: 0.1
Date: April 11, 2010

Author:

Anders Rundgren
Storbolsäng 50
74010 Almunge
Sweden

e-mail: anders.rundgren@telia.com

Appendix

The following XML extract shows how the Lookup Command is expressed in the mentioned implementation

```
<CredentialDiscoveryRequest  ...  >

   <LookupSpecifier ID="Lookup.1" Email="john.doe@example.com" Nonce="nSgmg4cznqE ... WrH2421w9SYA=">
      <ds:Signature>
         <ds:SignedInfo>
            <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
            <ds:Reference URI="#Lookup.1">
               <ds:Transforms>
                  <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                  <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
               </ds:Transforms>
               <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
               <ds:DigestValue>JBfoi8iBKRyWxXYlTTU1cdyybMTyJr+WDW+qCJdxoGE=</ds:DigestValue>
            </ds:Reference>
         </ds:SignedInfo>
         <ds:SignatureValue>mSMaH6wChPQRDT... JKrW3n/dL7seGbg==</ds:SignatureValue>
         <ds:KeyInfo>
            <ds:X509Data>
               <ds:X509IssuerSerial>
                  <ds:X509IssuerName>CN=Root CA,O=example.com,C=us</ds:X509IssuerName>
                  <ds:X509SerialNumber>2</ds:X509SerialNumber>
               </ds:X509IssuerSerial>
               <!-- The Issuer's Certificate: -->
               <ds:X509Certificate>MIIDbzCCAlegAw … gtzO/rITZcbKHyCZvQ==</ds:X509Certificate>
            </ds:X509Data>
         </ds:KeyInfo>
      </ds:Signature>
   </LookupSpecifier>

</CredentialDiscoveryRequest>
```

I.e. this command lookups all [EECERT] that having an e-mail address `john.doe@example.com` and are issued by the issuer listed in the `X509Certificate` element.