

Using Smart Crypto Modules to Improve Certification Authority Security

Dr. Stephen T. Kent
Chief Scientist- Information Security
BBN Technologies

Abstract

Much of the emphasis on CA security appears to focus on protecting the private key of the CA, a necessary but not sufficient aspect of secure CA operation. This presentation analyzes CA security from a system perspective, exploring a wide range of vulnerabilities that these systems often exhibit. This analysis shows that there are many ways to undermine CA security without compromising a CA's private key. An approach to CA system security that addresses many of these vulnerabilities, with minimal impact on most of the system, is presented.

The Problem

The fundamental security requirement for a CA is the accurate binding of a set of attributes in a public key certificate. Often we focus on ensuring the accuracy of just one attribute, i.e., the identity of entity that is represented as the Subject of a certificate. But, when certificates are used for access control purposes (vs. non-repudiation) the other attributes in a certificate may be equally or even more important than the Subject name. Viewed in this light, CA security requires that close attention be paid to ensuring that a CA sign a certificate (or CRL) only if it contains attributes that are accurate and consistent with CA policy, an important subset of the fundamental attribute accuracy requirement¹. Note that the need to protect a CA's private key is an obvious requirement in this light, since if that key is compromised, it can be used to sign certificates that contain attributes that are inconsistent with the policy of the affected CA.

In current CA systems, there are many points at which the attributes that will be bound to a public key can become inaccurate. A registration authority (RA) may be bribed to submit a certificate request with inaccurate attributes, or the workstation the RA uses may be attacked so that the request approved by the RA (displayed via some GUI) differs from what is submitted to the CA for signing. Personnel security failures can never be completely eliminated, but if the RA is required to sign each request, to provide a non-repudiable audit trail at the CA, this may deter malfeasance by an RA. It is unlikely that attacks against RA workstations will be prevented in most instances, because of the costs of providing high quality physical, procedural, and OS/application security for all RA workstations. The best one may be able to do in this regard is to constrain the certificate requests submitted by each RA. For example, if RAs are aligned with organizational boundaries, one can limit the range of subject names for which an RA can vouch.

Use of standard certificate request protocols (e.g., CMP or CMC) can counter attacks against the communication path between the RA and CA, so attacks against this portion of the system are readily addressed. Still, the CA end of the request path is vulnerable to many forms of attack. The computer on which the CA executes may be subject to attacks from the Internet, or close-in

¹ Some aspects of accuracy cannot be syntactically expressed, e.g., the association of a public key with the right individual.

attacks by maintenance personnel, subverted operators, etc. High volume, commercial CA service providers spend considerable amounts of money to afford good physical, personnel, and computer security for the CA facilities, staff, and computers, respectively. Still, these security efforts are not perfect, and thus there are non-trivial, residual vulnerabilities in the overall CA system. For example, the necessity to make use of commercial operating systems as a base for CA applications often limits the security quality of the CA platform itself. Moreover, personnel security is never perfect, so even commercial CA service providers are vulnerable to attacks of this sort. An organization acting as its own CAs will rarely be able to afford comparable personnel and physical security measures, and may not have the very best computer security either.

Since various components of a CA system may be subject to a very broad set of attacks, and given the primary CA security requirement articulated above, it is appropriate to ask whether there are components of the system where one can focus security attention to gain maximum benefit with minimum investment (with regard to both capital and life cycle costs). Ideally, one would like to identify a single component of a CA system that could ensure that all certificates (and CRLs) signed by the CA complied with CA policies. This is analogous to the goal of a network security administrator using a firewall at the periphery of a network to enforce site security policies, although one would like to achieve a much better security result here than we typically associate with firewall use.

Smart Crypto Modules

In fact, there is one component where one can implement high assurance security measures that will counter a wide range of attacks: the crypto module used by a CA. Since all certificates and CRLs are signed by the module, it is a “choke point” through which they must all pass, at least nominally. The crypto module is the last component in the CA system to process a certificate or CRL before it is issued². Thus any attack that would cause a certificate or CRL to violate CA policies, and which can be detected through examination of a “to be signed” certificate or CRL, could be thwarted by a “smart” crypto module.

The crypto module is an attractive point in the system on which to focus security attention, since it is already recognized as a critical security element with regard to protection of CA private keys. Good crypto modules offer better physical tamper resistance than any commercially available computer, better resistance to electromagnetic radiation attacks, etc. They don’t execute complex operating systems or vast quantities of application software, and thus are less vulnerable to a wide range of software-based attacks. In most cases these modules need not be “manageable” by CA operations staff, thus limiting opportunities for personnel security failures to circumvent the controls imposed by a crypto module. Finally, a CA system typically makes use of one or only a few crypto modules and thus additional costs associated with a more sophisticated version of this component are not multiplied in the same way as they would be for most other elements of the overall CA system where replication may be more common.

² Ex post facto checking after a signature is applied, e.g., before a certificate or CRL is posted to a directory, would not be as effective against all attacks. For example, a subverted CA staff member might introduce a “bad” certificate and have it signed, and then intercept it prior to posting, thus circumventing checks applied later in the process.

Almost all crypto modules used by CAs today are fairly simple devices. They are not capable of signing certificates or CRLs; rather, they sign what the CA workstation claims is a hash of a certificate or CRL. Such modules are not smart enough to meet the requirements noted above. Most of these modules were not designed specifically for CA support. Many corporate CAs use smart cards or PC cards that were originally designed to support an individual user; they have mediocre security and performance characteristics. Some CA service providers have moved to use higher performance crypto modules, but it appears that these modules have been adopted largely for performance reasons, or better base security features, i.e., the features mandated by FIPS 140-1 level 3 and 4. For the most part, these modules are not capable of acting as smart crypto modules.

The problem is not with the interface. The PKCS 11 interface employed by many crypto modules and supported by many CAs can support a “smart” crypto module, i.e., it embodies a “sign and hash” function that allows the object to be signed to be processed by the module. However, almost all crypto modules are “dumb;” they sign not the object, but a hash of the object, which often is computed outside of the module, to facilitate higher performance. One well-known, commodity CA does not even make use of an API for hashing, but instead embeds hash algorithm code in the CA and makes a call to the crypto API only for signing a hash.

It is possible to embody in a crypto module for a CA a considerable amount of sophistication that will significantly increase system security. As noted above, to achieve enhanced security, the module must be presented with the data object to be signed, e.g., a certificate or CRL, rather than just the hash of the object. Given access to the object, the module can examine it and apply syntactic checks to verify that the object structure and values are consistent with relevant aspects of the CA security policy. This directly supports the CA primary security requirement, by allowing the CA to enforce syntactic constraints on all the attributes as part of ensuring the accuracy of these attributes.

Syntactic constraints can be a very powerful means of ensuring, or at least improving, attribute accuracy for many of the certificate and CRL fields. In many instances, a CA might require that certain extensions, with specific values or with values chosen from a specified set, be present in end-entity certificates, while other extensions may be specifically prohibited. For example, a CA might require that the certificate policy extension be present and that it contain the value for the policy adopted by that CA. Often it will be appropriate to mandate the presence and values for key usage and extended key usage extensions. RFC 2459 mandates the presence of the authority key identifier extension in end-entity certificates, and the value of this extension should be taken from the subject key identifier extension in the issuing CA’s certificate.

A smart crypto module can ensure that the name constraints, policy mapping, and basic constraints extensions appear only in cross certificates, consistent with X.509 and RFC 2459 requirements. Cross certificates also must contain a basic constraints extension with the CA flag set to TRUE, another constraint readily enforced by a smart module, so long as the CA can distinguish between issuing end-entity vs. cross-certificates. These examples point out the need for the rules used by a smart crypto module to be able to express relationships among multiple fields in a certificate. X.509 and RFC 2459 mandate similar relationships among other fields (and field values) in certificates and CRLs.

Consider a more elaborate example: A CA might elect to allow some types of names to be specified in the name constraint extension, but may prohibit use of others. For allowed subject alternative name types, and in the subject name field of a certificate, a CA might impose constraints on the structure and value of the names represented. For example, a subject DN might be required to be subordinate (in the directory) to the issuer DN. There are many more examples of how suitable syntactic constraints can be imposed on certificates and CRLs to enforce CA security policies.

One may elect to define and enforce syntactic rules on a per-RA (not merely per-CA) basis. Doing so allows one to generate very precise rules which further support the requirement for ensuring attribute accuracy. An obvious example is to authorize RAs to approve certificate requests for well defined portions of the name space. So, for example, an RA in the Sprocket Division of Widget Corp. may request certificates only if the subject name is of the form “C = US, O = Widget Corp., OU = Sprocket Division” followed by a common name attribute. A smart crypto module should be able to enforce such constraints and a simple interface must be available to enable a CA manager to express such rules.

In order to achieve a high level of security through the use of smart crypto modules, the ability to create rules and bind them to CAs or RAs must be tightly controlled. Rule creation and binding should be a relatively infrequent event, and thus can be effected via offline means, without adversely affecting normal CA operation. For example, one could use a dedicated laptop to construct and sign rules, which can then be transferred to the crypto module in an authentic and integrity-secure fashion. This laptop would never be connected to a network and would be physically protected when not in use, to prevent introduction of malicious software. The private key used to sign the rules must also be carefully protected, e.g., using a crypto token. But, because the laptop and crypto token would be used infrequently and are physically small, it is feasible to afford good physical and procedural security at modest cost.

The smart crypto module associated with a CA needs to have the public key (from the rule signer) to verify the signature on the signed rules, and the introduction of that key into the smart crypto module must be tightly controlled. This suggests that a crypto officer role (a feature of FIPS 140-1 level 3 and 4 devices) be invoked to load this public key. If a crypto module supports multiple CAs, additional controls might be employed to ensure that the public key is associated with the right CA. To provide maximum benefits re personnel security, it must not be possible for the CA operational staff (with 24x7 access to the CA system components) to tamper with the rules that have been loaded or to load new rules. Since loading of new or modified rules should be a relatively infrequent event, appropriate procedural safeguards, based on separation of duties enforced via technical means, e.g., use of different crypto ignition keys (CIKs) can be implemented without adversely affecting operations.

As noted above, to achieve maximum security benefits one should associate rules with individual RAs and the module should be able to verify that a request emanates from a specific RA. This implies that a certificate request, in signed form, should be passed through the CA workstation, into the module, where the signature can be checked against an RA public key. One might choose to make RA certificates available to the module to perform this verification, but this introduces a sort of chicken-and-egg problem, i.e., who signed the RA certificate requests and why are they trusted to have done this correctly? In general it should be feasible to load

certificates (or equivalent signed structures binding RA name, CA name and public key) into the module relying on signatures generated by the same entity that signs the rules for the RA/CA. This represents a consistent authorization model that leverages the initial loading of the public key of the entity who signs rule sets, so that no additional manual intervention is required to load RA keys. Again, if the creation of new RAs, and the rate of RA key change is not too great, then this offline activity is easily managed without impinging on normal operation procedures.

SignAssure+

The crypto module functions described above, at the granularity of CA (but not RA-specific) rules, have been implemented in a product developed by BBN, designated SignAssure+. SignAssure+ is based on the latest BBN crypto module hardware platform³, which evolved from the first crypto module evaluated at FIPS 140-1 level 3. Its predecessor was also the first smart crypto module; it parsed certificates and CRLs prior to signing and enforced a small number of static constraints, but was not able to address the many extensions defined for version 3 certificates or version 2 CRLs. The SignAssure+ enforces rules defined using an ASN.1-based language, employed in a recursive fashion. It allows rules to be specified on a per-CA and a per-RA basis, although no facilities currently exist to process certificate request protocols (e.g., CMP or CMC) within the SignAssure+.

A GUI to facilitate rule construction, written in Java, executes on a (dedicated) laptop and rules are signed using a crypto module dedicated to that function. The interface tries to offer a consistent set of choices for the rule creator (require/permit/prohibit) whenever the semantics of the certificate and CRL fields and extensions permit. The public key for verifying signed rules is transferred on a CIK for direct loading to the SignAssure+, and is loaded under the control of the crypto officer and of the CA to which the rule key is bound. This satisfies the requirement that the operational staff cannot load or modify rules during steady-state CA operation, and it introduces two-party procedural control. A time-stamped, signed, internal audit trail, which is exported from the SignAssure+ under control of a separate role, provides additional protection against insider attacks. The SignAssure+ supports separate rule sets for end-entity and CA certificate signing, and for CRL signing, for each configured CA.

This sort of sophisticated crypto module does not address all aspects of the fundamental CA security requirement. For example, it cannot ensure that an RA is associating a certificate request with the specific individual named in the request. But it can significantly limit the damage that can result from accidental or intentional errors by an RA, from attacks against the RA's workstation, accidental or intentional errors by CA staff, and all forms of attacks against a CA workstation. By focusing security efforts on a limited, well-defined portion of the overall CA system, one can achieve a very attractive cost/benefit ratio, while dramatically lower the risks from a wide range of attacks.

³ This platform was previously known as SafeKeyper, a trademark now owned by Baltimore Technologies.