Evaluating Certification Authority Security

Stephen Kent
BBN Technologies
70 Fawcett Street
Cambridge, MA 02140
617-873-3988
kent@bbn.com

Abstract—A growing number of applications in the Internet are making use of X.509 public key certificates. Examples include security protocols such as SSL (used in web browsers), IPsec (used in firewalls and desktop computers), S/MIME (a secure e-mail protocol), and SET (the electronic commerce credit card transaction protocol). The public key certificates employed by the applications are created by Certification Authorities (CAs), that vouch for the binding of various attributes (e.g., identity) to a public key. Thus the security of these applications is dependent on the security of the CA function.

This paper examines security for CAs. It begins with a characterization of security requirements for CAs and continues with an exploration of the wide range of attacks that can be mounted against CAs. Included are attacks against network communications, against the operating systems used by CAs, "close-in" technical attacks against CA components (including cryptographic modules), and even misbehavior by human operators. The paper concludes with an examination of three approaches to implementing CA cryptographic support functions, analyzing each relative to the attack scenarios developed earlier in the paper.

TABLE OF CONTENTS

1. PUBLIC KEY CRYPTOGRAPHY AND CAS

Interest in the use of public key cryptography (PKC) is growing rapidly in the commercial and government marketplaces. In these contexts, PKC is commonly used either as a means for supporting management of symmetric keys for encryption, or for applying digital signatures to data. The cryptographic operations required to implement PKC can be implemented in software on a general purpose computer, or in dedicated hardware, e.g., PCMCIA or smart cards.

Most applications based on PKC technology make use of public key certificates (e.g., X.509 certificates) to facilitate distribution of public keys. The entity that applies a digital signature to a public key certificate is typically referred to as a certification authority or CA. Under X.509 certificate management standards [1], each CA also is responsible for maintaining a list of certificates issued, then revoked, by that CA. The relevant CA digitally singes its *certificate revocation list* (CRL).

In principal, the same software or hardware employed by users to provide cryptographic support for PKC signatures could be used by a CA for signing certificates and CRLs. However, the security and functional requirements for CAs differ from those for most end users, and thus it may be appropriate to adopt a different PKC implementation strategy for CAs. This paper examines security-relevant functions of CAs and the extent to which different technologies can be used to satisfy those requirements.

1.1 Basic CA Functions

As noted above, CAs perform two basic operations: issuing certificates and issuing CRLs. Figure 1 illustrates the relationships among elements in a typical certificate system. A user (User-1) interacts with a registration authority[1] (RA) to establish his identity, e.g., by presentation of credentials[2] such a passport or corporate ID card. The RA forwards the certificate request, including the user identity, public key[3] and other parameters, to the CA, via a secure communication path[4]. The certificate request is typically digitally signed, and optionally may be encrypted. The CA verifies that the request originated at an authorized RA, and applies any checks appropriate to that RA (or to all RAs) to the certificate request. After validating the request, the CA generates and signs a certificate, possibly supplying additional values for certificate fields based on some rule set (see below).

[1] Not all certification systems include a registration authority, but the use of an RA is increasinlg common. In the absence of an RA, a user may interact directly with a CA.
[2] On-line user registration is also feasible, especailly if the user is already known to the organization operating the CA, e.g., when the user is a client of a company which also operates a CA in support of an application.
[3] The user may generate his own public key pair, or it may be generated by the RA or CA. This example illustrates the first two cases.
[4] If this communciation takes place via a real-time protocol, the CA is termed "on-line." Alternatively, if the communication takes place via physical transfer of media, the CA is viewed as "off-line."

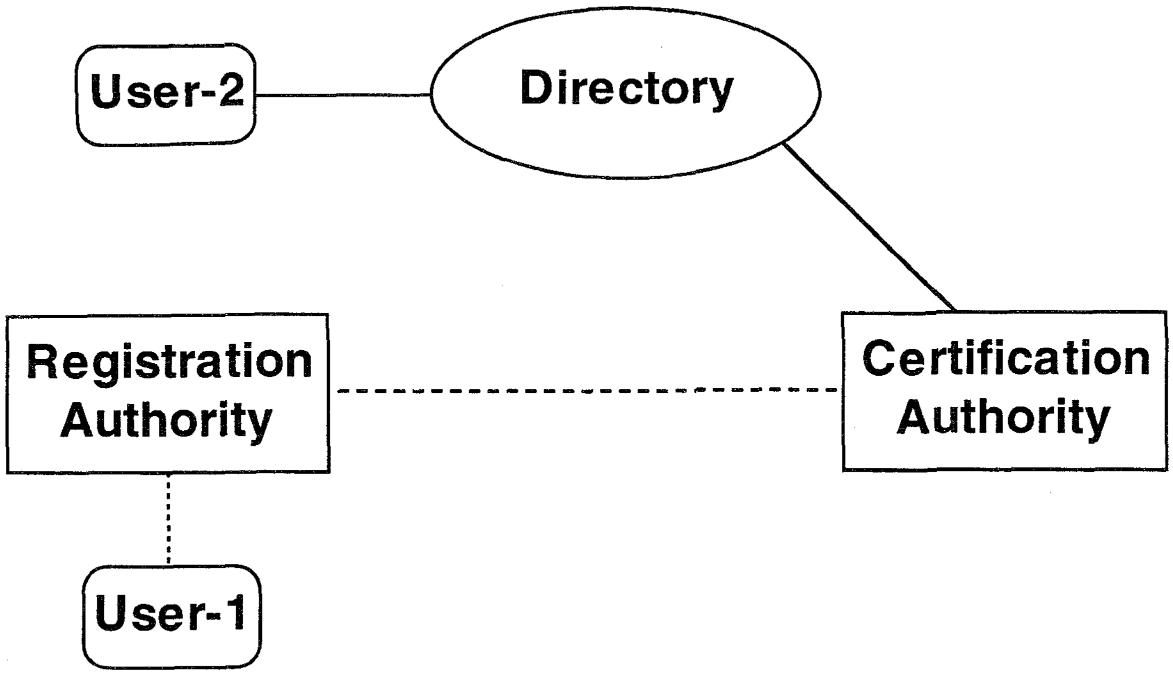

Figure 1: Typical Certification System Component Interactions

The resulting certificate is returned to the user in one of several ways, e.g., it may be sent back via the RA. In many instances, the CA will post the certificate to a directory. The CA also maintains a list of revoked certificates, called a certificate revocation list (CRL). This digitally signed list is posted to a directory on a periodic basis, determined by the CA. Users wishing to validate a certificate issued by a CA must check to ensure that the certificate has not been placed on the CRL. Users also may acquire certificates from a directory, or via direction interaction with other users.

First and foremost, in issuing a certificate, a CA[5] is expected to verify the accuracy of the identity of the subject whose public key is embedded in the certificate. Ultimately, this requirement must be satisfied by individuals, e.g., people function as CA or RA operators, and through appropriate procedures; it cannot be ensured through purely technical means. A similar requirement applies to the revocation function, where a person often is involved in ascertaining that it is appropriate to revoke a certificate in response to a request from a user or an RA.

However, a CA may be constrained with regard to the names that may be bound into a certificate, e.g., to prevent a CA from generating certificates for entities outside of the scope of the authority of the CA. These constraints help prevent errors by a CA and limit the damage that might be done by a rogue CA. These constraints can be enforced through technical means, e.g., via controls in CA software or hardware and via extensions in certificates themselves[6].

With the advent of X.509 version 3 certificates, there are many optional extensions that may appear in a certificate. Many of the "standard" extensions control certificate validation, or are used as inputs to authorization mechanisms. To the extent that these extensions are critical to secure use of a certificate, a CA should be as careful in ensuring the accuracy of this information as it is in verifying the accuracy of the subject name. Here too, individuals have the ultimate responsibility for verifying such information, whether the verification is effected directly by a CA or via a representative acting on behalf of a CA.

From a technical perspective, a CA security policy may require that some extensions be present, others must not be present, and others are truly optional. A required extension or optional extension may have a static value, an allowed set of values, or a set of prohibited values. There might even be some complex algorithmic constraints imposed on the values associated with groups of extensions. As above, these constraints can be enforced through technical means, to prevent errors and/or counter the threat of deliberate errors by individuals.

For CRL issuance, the basic CA requirement is to revoke certificates accurately, i.e., to revoke a certificate only as a result of a validated request consistent with the CA's policy for revocation. The X.509 version 2 CRL specification [1]

[5] In the example shown above, the RA assumes primary responsibility for authenticating the identify of the user, but the RA is viewed as an agent acting on behalf of the CA and so the distinction is ignored in this and later discussions.

[6] An X.509 version 3 certificate for a CA can contain an extension (NameConstraints) that constrains the range of names that should be viewed as certifiable by that CA.

adds optional extensions analogous to the version 3 certificate extensions. Here too, the policy of a CA may constrain the set of extensions that are employed, with some required, some prohibited, and others left as optional. For the required or optional extensions, various algorithmic constraints may be imposed on the values associated with these extensions.

This brief analysis shows that CA security requirements fall into two categories. Software and hardware cannot ensure that a CA does its job properly in terms of accurately identifying subjects, revoking certificates only when appropriate, or verifying the accuracy of data in certificate and CRL extensions. However, many other aspects of "correct operation" of a CA, e.g., algorithmic constraints on the content of certificates and CRLs, can be enforced through technical means. Enforcing such constraints through technical means protects against a wide range of errors and deliberate attempts to violate CA policy.

2. CA Security Requirements

Since a CA uses one or more private signature keys to sign certificates and CRLs, the most critical and obvious security requirement for CA operation is protection of these keys against disclosure. Disclosure of these keys would allow generation of apparently valid certificates that violate various aspects of a CA security policy. The most obvious attack resulting from disclosure of these keys is the issuance of certificates with inaccurate subject identification information, or with inaccurate certificate extension data. Generation of CRLs that revoke valid certificates, or omit previously revoked certificates, is another obvious attack based on key compromise.

The next section explores the sorts of attacks that might result in disclosure of CA signature keys, or that might allow issuance of erroneous certificates or CRLs via other means. However, in addition to the obvious security requirement for protection of CA private keys, there are other security-relevant requirements, some not so obvious, and these are discussed below.

For an individual user, destruction (not compromise) of a private signature key is an unfortunate event, but it is not a critical problem. Data signed with the key can be verified using the corresponding public key, in perpetuity. The user can acquire a new key pair and have a new certificate issued and continue with a relatively minor operational disruption.

In contrast, destruction of a CA (signature) key poses much more significant problems. Depending on how the CA fits into a certification system, distribution of a new CA public key may be very expensive and time consuming. The CA may loose its ability to issue CRLs for some time, causing security problems with respect to new revocation requests. Thus it is often important to be able to be able to backup and restore a CA private key, especially for a CA that acts as a "root" for a certification system (or for a substantial portion thereof). However, this recovery requirement must be performed in a fashion that does not undermine the confidentiality of these private signature keys.

Another set of CA security requirements arise from the demands of CA operational procedures. In many circumstances, more than one individual will be authorized to perform CA activities. This need certainly arises if a CA operates on a multi-shift basis, e.g., 24/7 coverage. This need also arises because individuals take vacations, sick days, etc. Individual accountability, i.e., a means of determining which individual was responsible for issuing a specified certificate or CRL, is an important requirement in this situation. Such accountability ensures that any problems with CA actions can be traced to the responsible party.

There is another way in which multiple individuals may act in the role of a CA, i.e., in parallel vs. serially over time. A CA policy may specify that more than one individual must approve issuance of a certificate or CRL, or of certificates with certain extensions, etc. The goal here is to prevent unilateral certificate management actions, rather than just relying on auditing and individual accountability as a remedial security measure. In such circumstances there may be a requirement to ensure that multi-party authorization is employed and to hold the responsible parties accountable for their actions.

CA operational requirements also may call for multiple CAs to be enabled simultaneously under the control of an individual, or group of individuals. For example, a company operating multiple CAs on behalf of other organizations may require that an operator be able to issue certificates under the auspices of multiple CAs with minimal overhead in switching among the various CAs. Even under these circumstances, the previously cited security requirements may come into play, i.e., individual accountability and multi-party control.

In some instances, it may be appropriate to polyinstantiate a CA, i.e., to cause the same CA private key(s) to be available at multiple locations. This may be a result of performance requirements or of the interaction of geographic and administrative domain boundaries. However, it is critically important that the procedure used to polyinstantiate CA private keys be secure, to prevent disclosure of these keys.

Finally, as noted in section 2, issuance of version 3 certificates and version 2 CRLs, may require constraining extension types and values. To the extent that these constraints can be expressed algorithmically, they are candidates for technical enforcement by CA software and/or hardware, rather than relying solely on CA personnel to correctly enforce these constraints.

3. Threats and Attacks

No security system is perfect. Viewed from a system perspective, every system contains some residual vulnerabilities. In deciding whether a given system is "adequately secure," it is necessary to establish a threat model, to provide a context within which to evaluate system security. A threat model identifies the set of adversaries that are perceived to be both motivated and capable of effecting attacks against a system. It also characterizes the attack capabilities of these adversaries. If none of the residual vulnerabilities can be exploited by a motivated adversary identified in the threat model, then the system is adequately secure.

CA operations, especially when they are generating certificates to be used in financial applications or in safeguarding access to sensitive corporate data, present attractive targets for attackers. The list of likely adversaries includes hackers, criminals, industrial spies, disgruntled employees, investigative reporters, terrorists, maybe even foreign intelligence agencies. These adversaries are differentiated in terms of their technical sophistication, resources, risk tolerance, and motivation. The following discussion explores some of these threats, setting the stage for evaluating the resistance of candidate CA security technologies to various forms of attack.

Figure 2 illustrates the functional elements of a typical CA system. The crypto module (CM) is the heart of the CA. It is where the private CA keys (used for signing certificates and CRLs) are generated and stored, and where signature operations take place. The CA workstation contains two critical databases, plus communication software and a management interface. The certificate and CRL database contains issued certificate and CRLs, and the rules database (RDB) defines certificate and CRL constraints. Registration Agents (RAs) interact with users to receive requests to issue or reissue certificates and to request certificate revocation. Communication with RAs is often provided via the Internet, through one or more firewalls, providing a means for receiving requests to issues certificates and CRLs, and for issuing signed certificates and CRLs. These security-critical elements of CA operation are illustrated below.

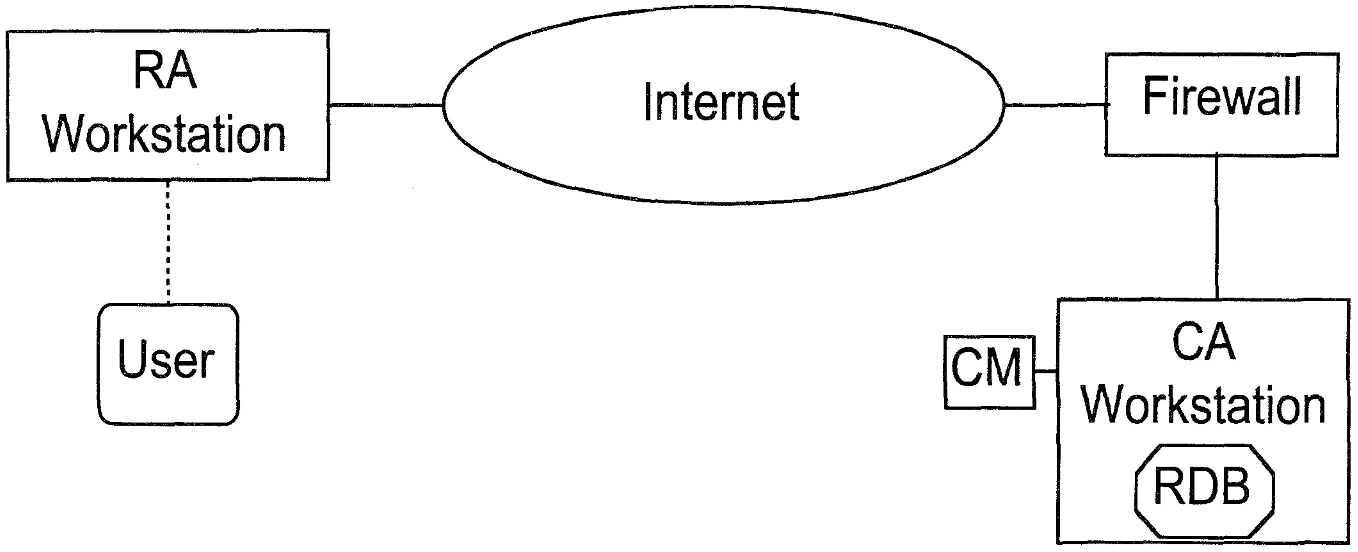

Figure 2: CA Functional Element Model

In the current environment, hackers typically effect attacks via remote access (which carries low risk of detection and apprehension), as opposed to attacks that entail physical proximity to the target or the use of invasive techniques. Hackers generally do not have extensive financial resources available, but sharing of attack software among hackers provides significant leverage. Relative to this threat, on-line CAs may be especially vulnerable. Even for off-line CAs, the communication paths used to interface with clients require careful protection, e.g., to avoid infiltration of malicious software. Note that a remote attack effected via a network need not entail trying to open a Telnet connection to a CA workstation and trying to guess a password. It could be more subtle form of attack exploiting operating system flaws to introduce malicious software, rather than attempting to gain real-time control of a CA. For example, RAs may communicate with a CAs via e-mail messages. This may leave the CA workstation vulnerable to any of the attacks that exploit various Sendmail bugs.

In contrast to the above scenario, several other classes of adversaries are much more likely to mount attacks that exploit physical proximity to the CA system. This includes attacks such as tampering with CA workstation hardware or software, crypto module hardware, even electromagnetic monitoring attacks. These attacks could be mounted directly by an external adversary or by disgruntled or bribed CA personnel. The more technical of these attacks are unlikely to be mounted against most CAs, but it is still instructive to examine the resistance of various CA technology to these attacks. It also is worth considering the adverse effects of non-malicious errors.

A careless individual operating in the role of a CA could inadvertently violate CA security policies. For example, he might issue a certificate with inaccurate subject name or extension data, perhaps as a result of an error in entering some data via a keyboard. It might seem an odd act to characterize benign errors of this sort as an attack, but experience shows that a large number of vulnerabilities in computer and network systems are the result of just such operational vulnerabilities. A malicious individual acting as a CA would typically have many opportunities to violate CA security, e.g., through introduction of intentional errors in the certificate data. In the absence of strong individual accountability measures, it might be impossible to determine which of several individuals authorized to act as a CA might have acted improperly, or simply carelessly.

If the CA is designed to control operator actions and/or to audit these actions (to prevent/detect some forms of abuse), there is still the possibility that an authorized individual with physical access to the CA might tamper with the software or hardware to override these controls. For example, an operator might reboot a workstation (from another medium) to circumvent controls employed by CA software in normal operation. Such an attack could also allow the operator to defeat audit mechanisms and thus allow him to conceal his actions. In addition to benign or malicious CA operator errors, there is always the potential for software or hardware bugs to adversely affect CA security. The recent experiences with faulty pseudo-random number generation in Netscape® and Kerberos (v4) [2,3] illustrates well this sort of problem.

If an adversary has physical access to the CA, he may be able to impersonate an authorized individual and invoke CA functions. With such access, CA software or hardware, even a hardware crypto module, could be covertly attacked. For example, an attacker with such access can introduce unauthorized software to capture any keys that become accessible during normal operation, or can add bogus certificate requests to the queue of valid one that will be processed when normal operation is resumed. Sophisticated forms of these attacks would evade many of the tools designed to detect software tampering on a workstation. Instead of actively tampering with hardware or software, the attacker might use physical access to plant sensitive monitoring equipment very close to the CA system, to extract information about the private keys. Some of these attacks are quite subtle, involving careful analysis of side effects associated with use of a private key for signing operations, e.g., monitoring timing, power consumption, or electromagnetic radiation from hardware.

More overt forms of attack also are possible. For example, a CA workstation or crypto module might be stolen, either to deny service to CA customers or provide an opportunity to employ invasive attack techniques to recover CA private keys.

4. Evaluating CA Implementation Options

A CA may be implemented in various ways, using a variety of hardware and software platforms and choices for crypto module support. This section examines several popular approaches to implementing a CA. In the cases examined below, we assume use of a COTS workstation platform and operating system (OS). The CA databases may entail custom software, or make use of a COTS database package. In contrast, the software implementing the core CA functions is specially developed for this purpose, even if the organization operating a CA does acquires the software (vs. developing it).

Use of a secure OS on the CA workstation can help address several of the types of attacks described in Section 3. If the CA operates in an on-line mode, it is potentially vulnerable to a wide range of attacks. Use of a secure OS can help counter some of these attacks. Operating system audit facilities can help detect attempts to tamper with software, though these facilities do not provide auditing at the granularity required for certificate and CRL issuance. Strong user authentication facilities can help deter unauthorized use of the CA, and contribute to the individual accountability requirement cited earlier.

The question naturally arises as to what constitutes an appropriate, secure OS for a CA. A number of COTS OSs have been evaluated at the C2 level by the National Computer Security Center (NCSC) under the Trusted Computer System Evaluation Criteria. However, most of these evaluations did not include network interface software, including the most recent, prominent example, Windows NT®. However, the C2 level of assurance is not especially stringent and those systems that were not evaluated with regard to network connectivity pose real questions for use in an on-line CA context.

Operating systems evaluated at the B1 or higher levels offer potentially greater security, although this potential may not be realized depending on how CA software is structured. The Compartmented Mode Workstations (CMWs) offered by several vendors are B1+ evaluated products including networking capabilities and windowing interfaces. These products represent good candidates. Still greater assurance might be had from products such as the SCC LOCK, which boasts A1+ security functionality[7]. Note, however, an adversary with physical access to the workstation still may be capable of circumventing many of the security provisions of any OS or application software.

4.1 Software Crypto Modules

The first approach we examine employs software to implement the crypto module element of the CA. Relative to FIPS 140-1 [4], most software crypto modules are likely to attain only level one compliance[8]. The Entrust® certification authority system is an example of such technology and it has been certified under FIPS 140-1. In this context, the crypto module usually maintains CA

[7] The LOgical Co-processor Kernel (LOCK) was designed and developed to very high assurance standards, but it has not undergone formal evaluation by the NCSC.

[8] It is possible to attain level 2 complinance under certain conditions. For example, the Netscape crypto module is level 2 compliant when used in the context of a CMW operating system under certain physical controls. However, this is a unique example, and not typical of how Netscape CAs are usually operated.

private keys in encrypted form when the CA function is not active. Each individual acting as a CA might employ a PIN or password to decrypt a copy of the keys[9] and enable the CA functions.

There are a number of limitations to use of a software crypto module. One obvious concern is that the module is limited to using a pseudo-random number generator (e.g., to generate the CA keys) and experience shows that this may result in subtle vulnerabilities. The integrity of the crypto module implementation is at risk, limited by the security of the OS and the physical security afforded the CA workstation

COTS workstations are not very resistant to tampering nor do they offer protection against attacks that exploit compromising electromagnetic emanations. Thus adversaries with physical access to the CA have many ways to subvert the security of this module through covert or overt physical attacks.

If CA keys are encrypted and stored on the workstation, then they may be susceptible to cryptanalytic attacks. The amount of entropy[10] available from a PIN or password is often not very great, so that recovery of the CA keys may be quite feasible, even if a strong encryption algorithm (e.g., triple DES) is employed. If the keys are stored off-line (e.g., on removable media), there may still be residues of the key in the workstation unless the operating system and CA application are carefully designed to avoid such problems.

Secure backup of CA private keys, and the related problem of CA polyinstantiation, is largely a matter of physical and procedural security in software-only crypto implementations. Probably the best that can be done for backup purposes, under these constraints, is to employ k-of-n secret sharing techniques [5] to split the private key values over a set of storage media (e.g., diskettes) that are stored in multiple, distinct locations. This technique would help ensure robust recovery in the event of failure of the CA workstation.

Ultimately, because the keys are present in the computer whenever the CA is active, malicious software infiltrated onto this workstation can capture the keys at that time, for later delivery to an attacker. This sort of attack can be effected despite the use of secret-sharing techniques, secure operating systems, etc. This may represent the most serious security limitation associated with any CA implementation that relies on a software crypto module. If there is any concern that personnel, physical, or procedural security may permit infiltration of such software, then this residual vulnerability may be unacceptable.

[9] If more than one individual can act as a CA, then each should have a different password, which can be used to encrypt a copy of the key which is used to encrypt the CA keys.

[10] In this context, entropy refers to the number of truly unpredictable bits in a secret bit string. For example, consider a password that is 8 characters in length. If the password is chosen from the space of lower case alphabetic characters, then the entropy is no greater than about 38 bits (vs. 64) and if one chooses from among English words (vs. random characters) the entropy is much lower, perhaps on the order of 15 bits.

4.2 Generic Crypto Modules: PC and Smart Cards

The advent of PC (formerly PCMCIA) and smart cards with cryptographic functionality creates an opportunity to move crypto module functions into hardware from software, minimizing some of the risks inherent in a software-only CA implementation. These cards are relatively low cost and are easily interfaced to a wide range of computers, providing an attractive alternative to software crypto modules. Examples of such modules are the Spyrus Links card and the Fortezza card (from Spyrus and Mykotronx), both of which are FIPS 140-1 compliant. No smart cards had been certified under 140-1 at the time this paper was prepared, but they would offer similar features, albeit with more limited memory and slower I/O interfaces.

A crypto card (e.g., PC or smart card) suitable for use as a CA crypto module generally includes the following features:

- hardware random number generator
- signature & hash algorithms
- storage of one or more private signature keys
- FIPS 140-1 level two secure packaging
- protection of keys via a PIN or password

The random number generator is important as a basis for having the card generate its own public key pair(s) internal to the card. The signature and hash algorithms are employed to sign certificates and CRLs using the private signature keys for the CA. Some cards may not have a hash algorithm available on the card, requiring the CA workstation to hash the certificates and CRLs prior to signing by the card. If the hash is computed externally, the card can't be sure what has been hashed, but unless the card is prepared to examine the syntax of certificates and CRLs, this distinction may not be critical.

To protect the card from undetected physical attacks, some form of tamper-evident packaging is appropriate. FIPS 140-1 provides a good reference for such packaging and level two compliance is feasible for these types of cards, and a few PC cards (specifically, the ones cited above) have been certified to this level of compliance. The use of a PIN or password to activate the card (typically to decrypt the private signature key) helps ensure that only authorized individuals can activate the CA.

The use of a PC or smart card as a hardware crypto module, counters some number of the attacks described in Section 3, but many others are still possible. For example, any CA security requirement that relies on syntactic checking of certificates and/or CRLs will still depend on correct operation of software on the workstation, because these cards do not have sufficient storage or processing capability to verify the constraints.

The size and packaging constraints imposed on these cards may make them susceptible to a number of attacks.

Proximate electromagnetic monitoring is likely still viable. The lack of an internal power supply may make these cards susceptible to timing channel [6] and power modulation attacks that have been described in the literature.

As noted above, tamper-evident packaging is probably no better than that defined for FIPS 140-1 level two. An overt attack, in which the card is stolen and subjected to destructive attacks at the convenience of the attacker, may be a valid concern with these cards. While various card and chip packaging techniques can be used to deter attacks, none is perfect. Published analysis in the last year, e.g., [7,8], has described methods by which a "tamperproof" card might be manipulated to discover private key information stored inside the card, without leaving evidence of tampering. This and other work suggest that it is prudent to assume that any CA private keys stored on the card can be recovered by a suitably sophisticated adversary, subject to the protection afforded by further cryptographic protection. However, PINs and passwords have limited entropy and thus are not ideal means of protecting CA private keys inside the card. In this context, keys on a stolen card may be susceptible to exhaustive search attacks, combined with attacks of the sort cited above.

The use of a PIN or password to enable the card represents a vulnerability in other respects. These short strings of bytes must pass through the CA workstation to be entered into the card. This makes them vulnerable to interception by malicious software on the workstation[11]. Also, PINs or passwords do not provide a good means of sharing responsibility among multiple individuals, e.g., for multi-party control, or of providing individual accountability.

To address the multi-party control problem, some cards (e.g., versions of the Spyrus Links card) now make use of split-key designs. In this technique, a private key used to sign certificates (or CRLs) is split among two or more cards, so that multiple cards must be used serially to effect the signature. However, to effect this split, one card is initially used to generate the CA's key pair, and then the private key is divided and distributed to the other cards. Thus a very high level of procedural, personnel, and physical security is required during this initialization phase.

Many crypto cards do not provide a secure means to transfer CA private keys in other cards. This facility is important as a means to protect against card failures that would result in loss of these keys, or to polyinstantiate a CA. The Fortezza card, developed for U.S. DoD use, does incorporate provisions for secure private key transfer. Like the split-key design approach described above, the Fortezza card relies on procedural and personnel security controls during the polyinstantiation initialization procedure.

[11] Staff at First Virtual wrote and demonstrated analogous software (that grabbed credit card numbers) in 1995, to bolster their arguments about the vulnerability of competing electronic payment schemes.

In summary, use of crypto cards as modules for CAs offers many advantages relative to software-based crypto modules, but a number or residual vulnerabilities remain.

4.3 CA-specific Crypto Modules

To achieve certification authority security better than that afforded by generic hardware crypto modules, e.g., PC cards, one must employ hardware specially designed to support CA functions. Such modules can incorporate all the advantage of generic crypto hardware, plus add CA-specific functions that help thwart the remaining vulnerabilities. To date only the BBN SafeKeyper® Signer [9], has been designed as a crypto module to exclusively to support CA operation, but other examples of CA-specific crypto modules may arise in the future. This section describes not only the features of the SafeKeyper, but also features one could imagine in such modules in the future, but which are present in no modules today.

Like the more capable generic crypto modules, any CA-specific crypto module should contain a hardware random number generator, storage for multiple sets of CA private signature keys, and a facility for hashing and signing certificates and CRLs. In addition to these facilities, the SafeKeyper also embodies some knowledge about the format of certificates and CRLs and thus it is able to verify some basic syntactic constraints on these data structures before signing them. However, the range of constraints that one might wish to express on certificate and CRL extension syntax is potentially very complex.

For example, for any extension, including a new privately defined extension, the CA might require that the extension be excluded, be presented with a specific value, be present with a range of allowed values, be optional with a default or range of values, or the presence and value range may be a function of the existence of other extensions! Thus it may not be practical to enforce all of these constraints within the boundaries of a CA-specific crypto module, while still preserving the simple software design that contributes to confidence in the operation of the device. SafeKeyper incorporates knowledge of X.509 v1 certificate and CRL formats and some knowledge of v3 certificate and v2 CRL formats, but additional knowledge could be incorporated.

In the future, it may be possible to load per-CA certificate and CRL constraint sets (the RDB in Figure 2) into a CA-specific crypto module, and have the module enforce these constraints, rather than relying on the CA workstation. This is attractive because these constraint sets can be constructed, verified, and signed off-line, by personnel other than those who provide direct CA operations. This separation of responsibility further minimizes opportunities for circumventing these constraints by tampering with the workstation. However, no modules available today embody this facility.

Because of differences in package design, the SafeKeyper device exceeds the FIPS 140-1 level three security requirements with regard to tamper-evident packaging, an improvement over what is available in PC or smart card crypto module formats. The SafeKeyper also exceeds these requirements in several other areas as well, e.g., it is designed to meet more stringent (classified) electromagnetic emanations and anti-tamper standards, providing better protection against a wider range of proximate, physically-based attacks. It also is resistant to the sorts of covert channel timing analyses alluded to in Section 3. Higher levels of security in these areas seem appropriate for CA crypto modules that might be subjected to sophisticated close-in attacks. Ideally, one would prefer FIPS 140-1 level four compliance in this regard, but no modules yet provide such security.

Even the size of a crypto module is security relevant, and it is here that PC and smart cards are at a disadvantage. Such small modules are easy to conceal on one's person, as part of a swap-out attack. This argues for bigger modules, ones can cannot be easily pocketed, yet there are disadvantages to very large modules too. For example, SafeKeypers are easily stored in a commonly available, multi-drawer, high security containers, whereas a large module (e.g., a 19" rack mount model) might require a much more expensive safe to hold it when not in use. Also, a module for which any surface is not completely visible and easily inspected for possible tampering is less than ideal, which argues against modules embedded in workstations on printed circuit cards.

Many generic crypto modules do a poor job of operator authentication, and only a few support the related notion of multi-party authorization, both of which are important aspects of CA support. Role-based authorization also is important for these modules. With role-based authorization, a single module can safely support multiple CAs, different roles can be supported for the same CA (e.g., certificate issuance vs. CRL issuance), and module management functions (e.g., creation or deletion of CAs) can be securely separated from per-CA operations. FIPS 140-1 requires some support for role-based authorization at higher evaluation levels, even for generic crypto modules.

Instead of employing PINs or passwords to identify individuals, the SafeKeyper makes use of crypto-ignition keys (CIKs[12]) and a sophisticated, internal key management technique. Each SafeKeyper contains a unique symmetric key[13], SK_{SYM} that is created by the device during initialization. When a CA is instantiated on a SafeKeyper, the private signature key for that CA (P_{CA}) is encrypted under SK_{SYM}. Then, this encrypted form of the private key is split between the device and a CIK, i.e., a random number is XORed with $[P_{CA}]^{SK}{}_{SYM}$, and the result is written to the CIK while the random number is held inside the SafeKeyper.

Each individual authorized to act as a CA is issued a personal CIK, which allows access to the corresponding CA private key. Use of CIKs provides much better security (vs. PINs or passwords) since the entropy is much larger and because the split data never passes through the CA workstation, but rather is entered directly into the crypto module. Thus attacks based on workstation interception of user authentication data are not viable in this context. If a SafeKeyper is stolen, and the CIK is not in the device, then even the most sophisticated overt attacks must break the symmetric algorithm (triple-DES) used to protect the CA private key.

To provide for multi-party control with redundancy, the SafeKeyper makes use of the Shamir secret-sharing algorithm, applied to the CIK split values. Each CIK contains one of the share values needed to activate a CA, allowing for k-of-n authorization. In the event that one or more of the CIK holders are deemed untrustworthy or the CIKs are lost, the remaining CIKs can be re-split, so long as the threshold number (k) of the CIKs can be brought together. After the re-split, any outstanding CIKs are no longer valid and pose no security danger for corresponding CA private key(s). With regard to role-based authorization, the SafeKeyper allows an individual to be authorized to issue certificates (or CRLs) for one CA or for multiple CAs. The authority required for various module management functions, e.g., configuring a SafeKeyper to add a new CA or to manage CIKs for a CA, is separable from per-CA operation authorization.

A critical feature for CA operation is fine-grained individual accountability, to detect who is responsible for issuing each certificate or CRL. Most CAs support this notion via CA workstation audit logs and by procedural and physical controls. However, given the vulnerabilities of workstations, and since it physical security may be poor in many private CA environments, it is preferable to enforce such accountability in the module. The notion is simple, i.e., the module could track internally the identity of the operator or set of operators who enabled CA operation. The resulting audit log could be output periodically in signed form under the control of a system security authority. Alternatively, the identity of the individual who issues a given certificate or CRL could be encoded into the certificate or CRL itself through the inclusion of a private (non-critical) extension for this purpose. With the addition of per-user IDs to CIKs, the SafeKeyper could support this feature, but it does not do so currently

As noted in Section 2, backup and polyinstantiation of CA private keys is critical, much more so than for users. This poses a dilemma for crypto module design, as any facility added to permit key polyinstantiation (which also serves the backup requirement) also has the potential to diminish system security. Some PC crypto modules (e.g., the Fortezza and Spyrus Links cards) support polyinstantiation, but with stringent requirements for procedural and personnel security. The SafeKeyper takes advantage of another aspect of the key management technology described above to support these requirements.

[12] A CIK is a key-shaped device that uses EEPROM to store from 1K-4K bits.

[13] In the initial model of the SafeKeyper, DES is used as the symmetric encryption algorithm, but later models make use of triple-DES to further deter cryptanalysis of a stolen device.

If it is necessary to clone a SafeKeyper, e.g., for polyinstantiation or backup purposes, the factory can produce a new device into which the symmetric key of the old device is transferred, in a secure fashion[14]. To transfer a CA's private key to this cloned device, a copy of the encrypted CA private key is output from the original device and input to the new device. When copies of the CIKs associated with that CA are inserted into the new device, the CA's private key is again made available for operations. This is a highly secure procedure; it requires the cooperation of the factory as well as those with direct physical access to the SafeKeypers, and at no time does either group of individuals have access to all of the keys required to gain access to CA private keys.

5. Summary & Conclusions

A number of disciplines are required to ensure security for a CA, including computer, cryptographic, procedural, physical, and personnel security. The range of attacks to which CAs may be subject is quite broad, including adversaries with either internal and external access and varying degree of technical sophistication. Most CA system designs to date have focused on protecting one aspect of CA security, private keys. Even so, the safeguards commonly employed do not afford very good protection in the face of possible insider attacks or attacks that might be effected by adversaries who gain close physical access to the computers or crypto modules used in CA systems.

Despite making use of the best technical security measures, there will always be some level of reliance on individuals to accurately identify the entities that are being certified, and to verify requests to revoke certificates. However, good designs can minimize reliance on many aspects of physical, personnel, procedural, and even computer security. Moreover, such designs can help ensure individual accountability, to deter individuals who might be tempted to act improperly if they could escape detection.

CA designs based exclusively on software are especially vulnerable, because of many security limitations intrinsic to software application environments and general purpose computers and operating systems. Use of crypto cards (PC or smart cards) as crypto modules can mitigate a number of the vulnerabilities of software crypto modules, but a number of other vulnerabilities remain. In particular, most of these cards are vulnerable to many forms of attacks that can be effected by those with physical access. Specialized hardware, designed specifically for use in the certificate and CRL issuance process, provides the best protection against the widest range of attacks, including one based on physical access to the hardware.

[14] When a SafeKeyper is initialized, it generates an RSA key pair and outputs the public portion of this pair. Each SafeKeyper also contains (in PROM) the RSA public key of a special "factory SafeKeyper." During initialization, the per-device symmetric key generated by the SafeKeyper is encrypted using this later public key and output for use in the cloning process. To make a new SafeKeyper made equivalent to an existing SafeKeyper the factory SafeKeyper is presented with the (encrypted) symmetric key from both the original and the new SafeKeypers. The factory SafeKeyper extracts both keys and encrypts the old symmetric key in the new symmetric key. The factory SafeKeyper then signs this encrypted token and presents it to the new SafeKeyper. The new SafeKeyper verifies the signature on this token and replaces its symmetric key with the symmetric key from the old SafeKeyper.

6. References

[1] ITU-T, Recommendation X.509, "Information Technology- Open Systems Interconnection- The Directory: Authentication Framework," June, 1997.

[2] Netscape (press release), "Beta Version of Netscape Security Update to be available Wednesday for free downloading," September, 1995.

[3] Information Week, "Security Flaw revealed in Kerberos," February, 26, 1996.

[4] NIST, Security Requirements for Cryptographic Modules, FIPS 140-1, January, 1994.

[5] Adi Shamir., How to Share a Secret," CACM 22 (11), November, 1979.

[6] Paul Kocher "Timing Attacks on Diffie-Hellman, RSA, DSA, and Other Systems," In N. Koblitz (ed.), *Advances in Cryptology - Crypto '96*, Lecture Notes in Computer Science, vol. 1109, pages 104-113, Springer-Verlag, 1996.

[7] Ross Anderson and Brian Kuhn, "Tamper Resistance - A Cautionary Note," Usenix Electronic Commerce 96, November, 1996.

[8] Eli Biham and Adi Shamir, "The Next Stage of Differential Fault Analysis: How to break completely unknown cryptosystems," Risks Digest, 18 (58), October, 1996.

[9] Charlie Gardiner, "Distributed Public Key Certificate Management" in Proceedings of the Workshop on Network and Distributed System Security (IEEE Press), February, 1993.

Stephen Kent is Chief Scientist- Information Security for BBN Technologies, part of GTE Internetworking, where he has worked on network and computer security projects for 20 years. He is a former and current chair of Internet standards (IETF) working groups, and chairs the Privacy and Security Research Group of the Internet Research Task Force. He has served on information system security committees for the National Research Council, the Office of Technology Assessment, and other government agencies. Dr. Kent has participated as a program committee member, panel or session chair for numerous security conferences. He received a B.S. in mathematics from Loyola University of New Orleans, and S.M., E.E., and Ph.D. degrees in computer science from MIT. He is a Fellow of the ACM.