A Closer Look at Revocation and Key Compromise in Public Key Infrastructures

David A. Cooper
Computer Security Division
National Institute of Standards and Technology
Gaithersburg, MD 20899

Abstract

Over time, in order to improve functionality or efficiency, new features have been added to the basic framework of public key infrastructures (PKIs). While these new features are useful, as with any other security critical application, new features can open the door for new types of attacks. In this paper, we will concentrate on those attacks against a PKI which allow an attacker to take advantage of a compromised private key. In particular, we will look at types of attacks that may allow an attacker, who has compromised someone else's private key, to either circumvent or exploit the mechanisms designed to deal with key compromise. The paper includes descriptions of several such attacks as well as suggestions to either prevent these attacks or to mitigate the damage that they can cause.

Keywords: public key infrastructure, certification authority, key compromise

1 Introduction

In any large scale public key infrastructure (PKI), there will be users whose private keys will be compromised. In order to mitigate the damage that a key compromise can cause, any certificates associated with a compromised key should be revoked. The purpose of revocation is to inform relying parties[1] that certain certificates should no longer be accepted as valid even though they have not yet expired. In general, the information provided to relying parties when a certificate is revoked (whether through certificate revocation lists (CRLs) [7], an on-line certificate status protocol (OCSP) [8], or some other mechanism) can be used in two ways. In addition to warning relying parties about certificates that should no longer be accepted as valid, many revocation schemes also provide information about the reasons that certificates were revoked. This information can be used, to a limited degree, to allow relying

[1] A relying party is anyone who uses the information in a certificate to make a decision.

parties to determine how to treat transactions that took place shortly before the revocation information was distributed.

The two ways in which revocation information may be used suggests two ways in which the revocation process could be manipulated. Once the certificates associated with a compromised key have been revoked, and information about the revocations has been disseminated, an attacker should no longer be able to impersonate the owner of the compromised key[2]. Thus, if an attacker were able to avoid being "locked out" by the revocation process, then we would say that the attacker had successfully circumvented the key compromise handling mechanism. Similarly, if an attacker were able to misinform relying parties about the reason that a certificate was revoked, then we would say that the attacker had successfully exploited the PKI's revocation mechanism.

In this paper, we will look at several potential attacks that either lead to the dissemination of false information about revocations or circumvent the revocation process. In section 3, we will present three attacks that deal with the reason codes which may be included in revocation announcements. Section 4 will describe an attack that may allow an attacker to circumvent the revocation process. In each of these sections, we will also present some ideas on how to design certification authorities in order to prevent, or at least minimize, the attacks.

2 An Overview of Public Key Infrastructures

The purpose of a PKI is to securely bind a set of attributes to a name. The name usually represents a person or a company, but may also represent a service (e.g., data storage or information retrieval) or even a component of the PKI itself (e.g., a certification authority (CA) or a registration authority (RA)). The set of attributes always includes the public key of the entity named in the certificate, but, usually, other information about the entity is made available as well.

The binding is accomplished through the use of certificates. A certificate is a data structure that includes an entity's name along with any information that is to be bound to that name. The entire certificate is signed by a CA. In order to be effective, the CA's public key must be well known (or be available through some secure mechanism) and the CA must be widely trusted.

The use of certificates is complicated by the possibility that information in the certificate will change. In order to enhance security, entities periodically change their public keys. In addition, people may change their names, jobs, or job titles. If a certificate contains attributes that are no longer valid, then the certificate should no longer be considered as valid.

As a general rule, entities change their public keys on a regular schedule. In order to support this in a clean manner, most certificates include expiration dates. The expiration

[2]In this paper, unless otherwise specified, we will only be considering digital signature certificates. However, attacks similar to the ones that we will describe may exist for other types of certificates.

date represents the time after which the CA that created the certificate is no longer willing to claim that the information contained in the certificate is valid.

Unlike the regularly scheduled change of public keys, name and job changes can not always be predicted far enough in advance to set the expiration dates on certificates correctly (certificates are frequently valid for a year or longer [6, 9]). Of even more concern, a user's private key may be compromised. A private key is considered to be compromised whenever it is in the possession of someone other than the key's owner (or someone trusted by the key's owner). Once a user's private key has been compromised, any certificate containing the corresponding public key should be revoked.

In order for CAs to invalidate (i.e., revoke) certificates before they expire, CAs must have some mechanism for distributing certificate status information. The two most common mechanisms for disseminating this information to relying parties are certificate revocation lists (CRLs) and on-line certificate status protocols (OCSP). A CRL is a signed data structure[3] that contains a list of unexpired certificates that have been revoked. In a CRL based system, updated CRLs are issued on a regular basis in order to allow relying parties to determine the current validity status of certificates. In an OCSP based system, relying parties send status requests to a trusted entity that responds with the current validity status of the certificate. It is expected that relying parties will send such requests to the OCSP server shortly before making use of the information in a certificate.

Certificate status protocols usually provide more information about certificates than just their current status (valid or revoked). For certificates that have been revoked, it is common to include some information about the reason for the certificate being revoked. For example, the **reasonCode** extension in X.509 version 2 CRLs [7] can be filled in with one of seven possible revocation reasons ranging from **keyCompromise**, representing the compromise or suspected compromise of the key, to **cessationOfOperation**, which simply means that the certificate is no longer needed for the purpose for which it was issued.

3 Reason Codes

In this section, we will look at three types of attacks that are based on the manipulation of certificate revocation reason codes. The first two cases involve attackers confusing relying parties by tricking certification authorities into providing the wrong reason code when revoking a certificate. In the final case, the attacker's manipulation of the reason codes leads to a delay in the dissemination of revocation information.

While some standards, such as X.509, specify a set of reason codes to be used when revoking certificates, we will not assume the use of any particular set of reason codes in this paper. Instead, we will group reason codes as necessary to describe the attacks.

A certificate can be revoked either for a benign reason (e.g., it is no longer needed) or because there is concern that the corresponding private key will be misused (e.g., key compromise). In this paper, we will use $\mathcal{B}$**enign** to refer to any code that implies a benign reason

[3] It is usually, but not always, the CA that signs the CRLs.

for revoking a certificate and $\mathcal{M}$**alicious** to refer to any code that could imply concern that the certificate will be misused. It should be noted, however, that, in practice, partitioning revocation reasons into those that are $\mathcal{B}$**enign** and those that are $\mathcal{M}$**alicious** might not be a simple matter. Looking at the X.509 reason codes, it is clear that **keyCompromise** and **cACompromise** should be classified as $\mathcal{M}$**alicious**. However, if the reason for revoking a certificate is **affiliationChanged**, the situation may not be as clear. If a subscriber's affiliation change is a result of being fired, then there is the chance that the subscriber will attempt to use his/her private key (until the revocation information is distributed) to cause damage to his/her former employer. On the other hand, most of the time when an employee changes jobs (whether within a company or by moving to another company) there is no cause for concern. From a security point of view, however, it may be prudent to treat **affiliationChanged** revocations as $\mathcal{M}$**alicious**. Similarly, prudence may require treating **unspecified** and **certificateHold** as $\mathcal{M}$**alicious** given the lack of information about the reason for the revocation.

The $\mathcal{M}$**alicious** reason codes can be separated into those that allow for repudiation of past actions and those that do not. Revocation reasons such as **keyCompromise** and **cACompromise** suggest that the subscriber may not be responsible for some of the messages signed in his/her name, particularly those signed shortly before the certificate was revoked. Other reason codes that could be classified as $\mathcal{M}$**alicious**, such as **affiliationChanged**, are classified as $\mathcal{M}$**alicious** to imply that the subscriber may attempt to request services that he/she is no longer authorized to request. This would not, however, suggest that someone other than the subscriber was signing the messages. In this paper, we will use $\mathcal{R}$**epudiable** to refer to reason codes which suggest that someone other than the subscriber may have signed messages using the private key associated with the revoked certificate, and $\mathcal{N}$**on**$\mathcal{R}$**epudiable** to refer to all other reason codes (whether $\mathcal{B}$**enign** or $\mathcal{M}$**alicious**).

3.1 Repudiating Malicious Actions

Whenever there is concern that a private key may be used to perform inappropriate actions (such as when key compromise is suspected), it is important to revoke the corresponding certificate as quickly as possible. Consequently, many certification authorities allow for on-line certificate revocation requests. If these requests are signed using the private key corresponding to the public key in the certificate to be revoked, then the CA can be set up to automatically accept such requests without opening the door to denial-of-service attacks.

When the private key corresponding to the public key in a certificate has been lost or stolen, the subscriber should request that the certificate be revoked with a reason code of **keyCompromise** (or some equivalent $\mathcal{R}$**epudiable** reason code). The $\mathcal{R}$**epudiable** reason code should suggest to relying parties that an attacker may have used the subscriber's private key to sign messages.

Unfortunately, in many cases, subscribers may have an incentive to lie about their keys being compromised since doing so might allow them to repudiate their own messages. For example, the owner of a certificate related to on-line credit card transactions may wish to

claim key compromise in an attempt to avoid paying for some recent purchases. As a result, requests for revocations with $\mathcal{R}$**epudiable** reason codes should not be taken at face value. So, in order for the $\mathcal{R}$**epudiable** reason codes to be meaningful, requests for revocation with $\mathcal{R}$**epudiable** reason codes should be investigated before they are accepted.

3.2 Masking Malicious Actions

Just as legitimate users may attempt to manipulate the use of reason codes in certificate revocation systems, so may attackers. While a subscriber may attempt to repudiate messages that he/she actually signed, an attacker may attempt to prevent a subscriber from repudiating messages that the attacker signed in the subscriber's name. One way an attacker could do this would be to request that the certificate be revoked with a $\mathcal{N}$**on**$\mathcal{R}$**epudiable** reason code attached shortly after signing some bad messages. If the reason code used was also $\mathcal{B}$**enign** then relying parties would have no reason to suspect the messages that were signed by the attacker.

3.3 Manipulating Segmented CRLs

It has frequently been suggested that CRLs should be segmented by reason code with those CRLs representing $\mathcal{M}$**alicious** reason codes being updated more frequently than those representing $\mathcal{B}$**enign** reason codes [6, 7]. For example, the CRLs representing $\mathcal{M}$**alicious** reason codes could be updated daily (or several times a day) while the CRLs representing $\mathcal{B}$**enign** reason codes could be updated weekly or even monthly [1, 9]. Ideally, such an arrangement would reduce communication costs and increase the utility of CRL caches without affecting security.

Unfortunately, the disparity in update frequencies between CRLs for $\mathcal{B}$**enign** and $\mathcal{M}$**alicious** reason codes could open the door for an attacker. Suppose that an attacker compromises a subscriber's private key and then submits a revocation request with a $\mathcal{B}$**enign** reason code. While this might seem counterproductive, if the CRL corresponding to the reason code used is only updated monthly, then the attacker could be free to use the compromised key for up to a month. Furthermore, if the certification authority was not carefully designed, the $\mathcal{B}$**enign** revocation request from the attacker could prevent the subscriber from getting the certificate revoked for a $\mathcal{M}$**alicious** reason. In other words, the attacker's revocation request could block future attempts to get the certificate placed on a frequently updated CRL.

A similar outcome could occur if the attacker were to compromise a private key, or began to use a compromised private key, after the corresponding certificate had been revoked for a $\mathcal{B}$**enign** reason. Even though the certificate had already been revoked before the attacker first used the private key, signatures created using the key would still be accepted as valid until the revocation information had been distributed.

3.4 A Proposed Solution

As described in sections 3.1 and 3.2, revocation reason codes provided by subscribers, especially those provided through on-line transactions, can not always be accepted at face value. Therefore, in order for the reason codes attached to certificate revocation announcements to be meaningful, CA administrators should not publish revocations with reason codes other than **unspecified** unless the reason for the revocation has been substantiated. Of course, the amount of effort expended to substantiate the reason code information should be based on the amount of damage that could be caused as a result of incorrect reason code information being distributed.

Whenever there is concern about malicious behavior, it is important to revoke certificates as quickly as possible. So, whenever a subscriber (or other authorized party) requests that a certificate be revoked with a $\mathcal{M}$**alicious** reason code, the certificate should be revoked immediately. If a substantial amount of time is needed to substantiate the claimed revocation reason, then the certificate should be revoked with a reason code of **unspecified**. Since revocation reason codes may be useful to relying parties, the CA software should be designed to allow the CA administrator to change revocation reasons. The ability to change reason codes for a revoked certificate should also be available for when malicious behavior is discovered after a certificate has been revoked for a $\mathcal{B}$**enign** reason. After a reason code is changed, the certificate's revocation announcement should be redistributed in a manner that is appropriate for the new reason code.

While the basic idea behind CRLs segmented by reason code is useful, maintaining such CRLs may be difficult in an environment in which the reason codes for revocations may be changed even after the original revocation information has been distributed. Instead, we suggest using a slight variant on the idea of Δ-CRLs. Suppose that one wishes to distribute revocations with $\mathcal{M}$**alicious** reason codes daily while only distributing those with $\mathcal{B}$**enign** reason codes once a month. Then, once a month, a full CRL should be issued containing all unexpired certificates that have been revoked for any reason. Once each day, a Δ-CRL should be issued which contains all unexpired certificates that have been revoked for a $\mathcal{M}$**alicious** reason that do not appear on the most recently issued full CRL. Thus, a relying party only needs to look at two CRLs, one full CRL and one Δ-CRL, in order to determine the status of a certificate. As long as CA administrators can change the reason code for a revoked certificate, the attack described in section 3.3 will not be possible.

This scheme has two main advantages over CRLs segmented by reason code. First, since Δ-CRLs only contain the most recently revoked certificates, they are kept relatively small. Thus, communication overhead is reduced. This technique also avoids the problems that could occur from allowing revocation reason codes to change. In a system with CRLs segmented by reason code, CRLs should be scanned in the order in which they were issued (most recent first) in order to ensure that the most up-to-date revocation reason is obtained. In the Δ-CRL approach, any revocation information in the most recently issued Δ-CRL should be considered more recent than revocation information in the most recently issued

full CRL[4].

4 Circumventing Revocation through On-line Renewal

When an entity suspects that an attacker has compromised its private key, it will request the revocation of any certificate containing the corresponding public key. Ideally, once information about the revocation of these certificates has been distributed, no future attacks based on the key compromise should be possible. The expiration of a certificate should have a similar effect. Even if a key's compromise is never detected, no attacks based on the key compromise should be possible once all of the certificates containing the corresponding public key have expired. As we will show in this section, achieving this goal may not be a trivial matter.

The problem that we will describe in this section can occur in CAs that allow subscribers to use proof-of-possession of the private key corresponding to the public key in one certificate as proof of identity for the purposes of obtaining another certificate. The most well known example of this is on-line renewal. In on-line renewal, a subscriber uses the private key corresponding to the public key in a certificate that is about to expire in order to obtain a new certificate of the same type with a later expiration date (and usually a new public key). In principle, however, a subscriber should be allowed to use one certificate to request a new certificate of a different type. For example, a subscriber could use the private key associated with a signature-only certificate to sign a request for a key management certificate. Allowing such a self-signed certificate request would save both the CA and the subscriber from repeating the identification process involved in an initial registration.

In this section, we will consider both on-line renewals and general self-signed certificate requests. As we will demonstrate, allowing general self-signed requests makes the handling of key compromise and other $\mathcal{M}$**alicious** revocations much more difficult, particularly when the CA issuing the new certificate is different from the CA issuing the original certificate. So, we will first cover the limited case of on-line renewals and then discuss the general case.

4.1 On-line Renewals

Let CA be a certification authority that has issued a certificate which binds user A to A's public key, K_A, and let B be an attacker which has learned the private key that corresponds to K_A, K_A^{-1}. Using K_A^{-1}, B can send a certificate renewal request to CA containing a new public key, K_B, corresponding to a private key, K_B^{-1}, that only B knows.

B's ability to perform an on-line renewal leads to several complications. First, since A will be the subject of the new certificate, A will need to get this certificate revoked in addition to the original certificate in order to prevent B from impersonating A once the key compromise

[4]If the reason for revoking a certificate is changed from a $\mathcal{M}$**alicious** reason code to a $\mathcal{B}$**enign** reason code then the certificate's serial number, which at one point appeared on a Δ-CRL, may temporarily not appear on any CRL. It will reappear on the next full CRL to be issued. During the period in which the serial number does not appear on any CRL, some relying parties may accept the certificate as valid.

has been discovered. Furthermore, since the new certificate will have an expiration date later than the old certificate (and since B may be able to perform an on-line renewal using the new certificate), B's ability to impersonate A will far outlive A's original certificate. Therefore, if A does not discover the key compromise, B will be able to impersonate A long after A's original certificate expires.

There are several things that can be done to help minimize the problems associated with on-line renewal. First, a CA may allow each certificate to be used for on-line renewal at most once. Since an on-line renewal request is only used to request a certificate of the same type as the certificate used to make the request, this restriction should not pose any problems for legitimate users. This restriction has two main advantages. First, if a legitimate user attempts to perform an on-line renewal then it will either be successful or it will fail. If it is successful, an attacker that has compromised the private key will be unable to perform an on-line renewal. So, even if the compromise is not detected, the attacker's ability to impersonate the user will end once the original certificate expires. If, on the other hand, the legitimate user is unsuccessful in performing the on-line renewal as a result of the attacker having previously performed an on-line renewal, then the legitimate user will discover the key compromise and can begin the revocation process.

In addition to limiting the number of on-line renewals that can be performed, a CA can also limit the times in which on-line renewals will be accepted. Since on-line renewals are generally used to replace certificates that are about to expire, it should be reasonable for a CA to only accept on-line renewals during the period of time shortly before the original certificate expires. The main advantage to this is that it will prevent the attacker from performing a sequence of renewals which could result in a long, and perhaps difficult to follow, chain of certificates.

Whenever a CA allows for on-line renewal, the possibility of an attacker possessing improperly obtained certificates must be taken into account when revoking a certificate (particularly when the reason for the revocation is $\mathcal{M}$**alicious**). As a general rule, whenever a certificate has been revoked for a $\mathcal{M}$**alicious** reason, any certificates created through an on-line renewal based on that certificate should also be revoked. If the revocation reason is $\mathcal{B}$**enign** then it may not be considered necessary to revoke the new certificates. However, it should be remembered that the revocation request could have come from an attacker. So, if the new certificates are not revoked, the CA should be set up in such a way that the new certificates can be found and revoked if it is later determined to be necessary.

The situation becomes more complicated when the owner of a certificate does not attempt to renew or revoke the certificate. In many systems, some subscribers who no longer wish to have a certificate will choose to just let their current certificates expire. In this case, it would be very easy for an attacker that has compromised a private key to perform an on-line renewal without ever being detected. While there is no perfect solution to this problem, many things can be done to at least give the subscriber a reasonable chance of discovering that someone else has used its certificate to perform an on-line renewal.

The simplest technique available to the CA is to publish all certificates in a repository. While there is no guarantee that the subscriber will take the precaution of checking the

repository for any new certificates with itself as the subject, the CA has at least made the information available. Similarly, the CA could revoke all certificates that have been used in an on-line renewal with a reason code of **superseded** (or equivalent). Either of these techniques may be considered acceptable as long as certificates and/or CRLs are easily accessible to former subscribers.

While the above techniques may be acceptable in many systems, they generally involve the active participation of users who are no longer subscribers. Given this, one may be more successful by sending renewal notifications directly to the subscribers using either electronic or paper mail. While such notifications could be ignored, or even intercepted by an attacker, they may be more successful than the repository technique since less effort is involved on the part of the former subscriber. Of course, if the notification is in the form of a charge on a credit card statement, then the subscriber has an even greater incentive to deal with inappropriate renewals.

4.2 General Self-Signed Certificate Requests

Unfortunately, many of the suggestions for handling problems with on-line renewal do not apply to the more general case of self-signed certificate requests. In the general case, it is unreasonable (and perhaps impossible) to limit the number of certificates that a subscriber can request using a certificate. Similarly, limiting the times during which a certificate can be used for certificate requests (except to require that the certificate be valid at the time of the request) would not make sense.

While the lack of restrictions on the use of self-signed certificate requests limits the chances that an attacker's actions will lead to its discovery, the CA should still be set up to properly handle revocations for the cases in which the key compromise is discovered (or for whenever a certificate is revoked for a $\mathcal{M}$**alicious** reason). As such, for each certificate, a CA should maintain a list of all certificates that it issued that were created as a result of a self-signed certificate request based on the certificate. Then, whenever a certificate is revoked for a $\mathcal{M}$**alicious** reason, all of the certificates created based on that certificate should also be revoked.

As we mentioned in section 4.1, a CA should also be designed to prevent an attacker from taking advantage of a compromised key once the corresponding certificate has expired. There are two ways that this property can be carried over to the general case. The straightforward technique would be to always give the new certificates expiration dates that are no later than the expiration dates on the certificates used in the requests. This solution particularly makes sense in a system that does not allow for on-line renewal. The alternative is to use one of the techniques described in section 4.1 in order to attempt to notify the subscriber that a new certificate has been created.

It is not unreasonable to expect that some CAs will allow subscribers to use certificates issued by other CAs in order to prove identity when requesting a certificate. For example, a certified private signature key could be used by a subscriber to sign requests for identification and key management certificates. In many cases, the CA issuing the identification and key

management certificates will be different from the CA that issued the signature certificate.

While there may be sound business reasons for allowing this type of on-line certificate request, it does significantly complicate the handling of key compromise. Consider the case in which a subscriber, A, has a certificate, $CERT_A$, issued by CA_A. Suppose that an attacker has compromised A's private key and has used it, in conjunction with $CERT_A$, to obtain a certificate, $CERT_B$, from CA_B. Even if A discovers that its key has been compromised, it may not know that B has obtained a new certificate, with A as the subject, from CA_B. Some of the techniques described in section 4.1 for informing A about the certificate request may still be viable in this environment. However, CA_B can not expect A or CA_A to look in its repository since A and CA_A may not even know that CA_B exists, let alone that it accepts certificates issued by CA_A for proof of identity. So, unless CA_B obtained A's address (either electronic or physical) from CA_A, CA_B must take responsibility for keeping track of the status of $CERT_A$.

When CA_B accepts a certificate request based on $CERT_A$, CA_B is, in effect, acting as a relying party with respect to $CERT_A$. Viewed this way, the key compromise handling problem can be dealt with if CA_B follows two rules. First, the expiration date on $CERT_B$ should be no later than the expiration date on $CERT_A$. This will prevent B from exploiting the key compromise beyond the expiration of $CERT_A$. Second, CA_B should periodically check the revocation status of $CERT_A$ (through CRLs, OCSP, or some other means). If CA_B discovers that $CERT_A$ has been revoked, then it should revoke $CERT_B$ (probably using the same reason code).

5 Conclusion

As we have shown in this paper, dealing with revocation and key compromise in a secure manner can be very complicated. We have presented some of the potential problems that may arise along with suggestions for handling these problems. However, in any particular system, there may be yet other problems and the optimal solutions to these problems may differ from those suggested in this paper.

One major area that still needs to be addressed is the interpretation of reason codes. As we said earlier, determining whether a reason code should be classified as $\mathcal{B}$**enign** or $\mathcal{M}$**alicious** is not a trivial matter. To whatever degree is possible, groups working to define sets of reason codes should clarify the reason codes' meanings in order to enable the proper handling of revocation requests.

Acknowledgements

We would like to thank Bill Burr, Donna Dodson, Nelson Hastings, Noel Nazario, and Tim Polk who provided many helpful discussions on this research.

References

[1] Shimshon Berkovits, Santosh Chokhani, Judith A. Furlong, Jisoo A. Geiter, and Jonathan C. Guild. *Public Key Infrastructure Study: Final Report*. Produced by the MITRE Corporation for NIST, April 1994.

[2] Marc Branchaud. A survey of public-key infrastructures. Master's thesis, McGill University, March 1997.

[3] Santosh Chokhani. Toward a national public key infrastructure. *IEEE Communications Magazine*, 32(9):70–74, September 1994.

[4] Warwick Ford. Advances in public-key certificate standards. *ACM SIGSAC Security Audit & Control Review*, 13(3), July 1995.

[5] Warwick Ford. *A Public Key Infrastructure for U.S. Government Unclassified but Sensitive Applications*. Produced by Nortel and BNR for NIST, September 1995.

[6] Warwick Ford and Michael Baum. *Secure Electronic Commerce: Building the Infrastructure for Digital Signatures and Encryption*. Prentice Hall, 1997.

[7] Russell Housley, Warwick Ford, William Polk, and David Solo. *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. IETF X.509 PKI (PKIX) Working Group, June 1998. (draft).

[8] Michael Myers, Rich Ankney, Ambarish Malpani, Slava Galperin, and Carlisle Adams. *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. IETF X.509 PKI (PKIX) Working Group, June 1998. (draft).

[9] VeriSign. VeriSign certification practice statement (CPS), May 1997. Version 1.2.