C. Adams(Entrust Technologies)
R. Zuccherato(Entrust Technologies)

Internet Draft
expires in six months

June 4, 1998

Data Certification Server Protocols

<draft-adams-dcs-00.txt>

Status of this Memo

Abstract

This document describes a general data certification service and the protocols to be used when communicating with it. The Data Certification Server is a Trusted Third Party (TTP) that can be used as one component in building reliable non-repudiation services (see [ISONR]). Useful Data Certification Server responsibilities in a PKI are to validate signatures and to provide up-to-date information regarding the status of public key certificates. We give examples of how to use the Data Certification Server to extend the lifetime of a signature beyond key expiry or revocation and to query the Data Certification Server regarding the status of a public key certificate.

> The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in RFC 2119 [RFC2119].

1. Introduction

A Data Certification Server (DCS) is a Trusted Third Party that verifies the correctness of specific data submitted to it. The Data Certification Server provides the data certification service in order

that non-repudiation evidence may be constructed relating to the validity and correctness of an entity's claim to possess data, the validity and revocation status of an entity's public key certificate and/or the validity and correctness of another entity's signature. When certifying possession of data or another entity's signature, the DCS

verifies the mathematical correctness of the actual signature value contained in the request and also checks the full certification path from the signing entity to a trusted point (e.g., the DCS's CA, or the root CA in a hierarchy). The DCS MAY be able to rely on all relevant CRLs and ARLs, or the DCS MAY need to supplement this with access to more current status information from the CA. It then includes a trusted time and creates a data certification token. (See Appendix B.)

When certifying a public key certificate, the DCS verifies that the certificate included in the request is a valid certificate and determines its revocation status at a specified time. Again, it checks the full certification path from the certificate signing entity to a trusted point. The DCS MAY be able to rely on all relevant CRLs and ARLs, or the DCS MAY need to supplement this with access to more current status information from the CA. It includes this information, along with a trusted time, to create a Data Certification Token. (See Appendix C.)

The presence of a data certification token supports non-repudiation in two ways. It provides evidence that a signature or public key certificate was valid at thetime indicated in the token. The token can be used even after the corresponding public key certificate expires and its revocation information is no longer available on CRLs (for example). The production of a data certification token in response to a signed request for certification of another signature or public key certificate also provides evidence that due diligence was performed by the requester in validating the signature or public key certificate.

It is not recommended that the DCS be used as a substitute for normal public keycertificate revocation checking (e.g. CRLs, OCSP) in large environments, due to concerns about the scalability of this protocol. It should only be used to support non-repudiation or to supplement more traditional revocation services when more timely information is required.

In all cases, the trust that PKI entities have in the Data Certification Server is transferred to the contents of the data certification token (just as trust in a CA is transferred to the public key certificates that it issues). As a particular example, a data certification token pertaining to a signature may be useful for extending the life of that signature beyond the expiry or subsequent revocation of its corresponding verification certificate.

2. Requirements of the Data Certification Server

The Data Certification Server is REQUIRED to:

1. verify the correctness of the enclosed digital signature using all appropriate status information and public key certificates and produce a signed data certification token attesting to the validity of the signature, if asked by the requester.
2. verify the validity (according to [CCP]) of the enclosed public key certificate and its revocation status at the specified time using all appropriate status information and public key certificates and produce a signed data certification token attesting to the validity and revocation status of the public

key certificate, if asked by the requester.
3. include a monotonically incrementing value of the time of day or a time stamp token into its data certification token.
4. include within each signed data certification token an identifier to uniquely determine the trust and validation policy used for this signature.
5. sign each data certification token using a key generated exclusively for this purpose and have this property of the key indicated on the corresponding public key certificate.
6. indicate in the token whether or not the signature or public key certificate verified, and if not, the reason the verification failed.
7. provide a signed receipt (i.e., in the form of an appropriately defined data certification token) to the requester, where appropriate, as defined by policy.

3. Data Certification Server Transactions

As the first transaction of this mechanism, the requesting entity requests a certification by sending a request (which is or includes a data certification request, as defined below), including the data for which validity and/or possession is to be certified, to the Data Certification Server. Upon receiving the request, the Data Certification Server reviews and checks the validity of the request. A valid request is of the form described in Section 5 of this document, can be properly decoded, and is from a supported Data Certification Server subscriber. If the request is valid, the Data Certification Server performs the certification and sends a response (which is or includes a data certification token, as defined below) to the requesting entity. Otherwise, the Data Certification Server returns an error message (i.e., in the form of an appropriately defined data certification token).

Upon receiving the token, the requesting entity verifies its validity. The requester SHOULD verify that it contains the correct time, the

correct name for the DCS, the correct data imprint, a valid signature, and satisfactory status, service and policy fields. Since the DCS's certificate may have been revoked, the appropriate status information SHOULD be checked to verify that the certificate is still valid. The token can now be used to authenticate the correctness or possession of the corresponding data.

4. Identification of the DCS

The DCS MUST sign all data certification messages with a key reserved specifically for that purpose. The corresponding certificate MUST contain the extended key usage field extension as defined in [CCP] Section 4.2.1.14 with KeyPurposeID having value id-kp-dcs. This extension MUST be critical.

```
id-kp-dcs    OBJECT IDENTIFIER ::= {id-kp  ??}
  -- Certifying the validity of certain information.  Key usage bits
  -- that may be consistent:  digitalSignature, nonRepudiation
```


5. Request and Token Formats

The ServiceType type indicates which type of Data Certification Server Service is required.

```
ServiceType ::= INTEGER  { cpd(1), cs(2), cpkc(3) }
```

The value cpd (Certify Possession of Data) is used when only the signature on the data certification request (i.e., possession of the data in the request) is to be verified. In this case the Data Certification Server would be merely providing evidence that the requester possessed the data in the request and a valid signature key at the time indicated. This is really an extension of the Time Stamp Authority [TSA] in that we are given the additional assurance about the validity of the signature, as well as the time before which it was applied. The value cs (Certify Signature) is used when another entity's signature is to be validated. The resulting token can then be used to support non-repudiation services or to allow use of the signature beyond public key certificate revocation or expiry.

The value cpkc (Certify Public Key Certificate) is used when the validity and revocation status of the public key certificates included in the request are to be verified. This service can be used to supplement the use of CRLs when timely information regarding a certificate's revocation state is required (e.g. high value funds transfer or the compromise of a highly sensitive key) or when evidence supporting non-repudiation is required. A given DCS MAY support any subset of the above services.

Upon receiving a signed request for either service cs or cpkc the DCS MUSTalso verify the signature on the request as is done for the cpd service. Note however, that signed requests for the cs or cpkc service are not required.

A data certification request is as follows. It is encapsulated as a SignedData construct [CMS]. The content is of type DCSReqData, which is indicated by the OID:

```
DCSReqData OBJECT IDENTIFIER  ::= { ??????  }
```

The data certification request MUST either be unsigned or contain only the signature of the requester.

The data and information that will be certified are contained in the content field of the SignedData content type.

```
DCSReqData ::= SEQUENCE  {
     dcsReqInfo                DCSReqInfo,
     data                         Data
       --the data to be certified
       --this field MUST be of type Message if the service type is cs
       --and of type SEQUENCE OF Certificate if the service type is cpkc
}
```

The dcsReqInfo field contains information pertaining to the data certification request.


```
DCSReqInfo ::= SEQUENCE  {
     version                      Integer { v1(0) },
     service                      ServiceType,
     requester                    GeneralName OPTIONAL,
      --MUST be present if the service field is cpd
      --MUST match the identity (subjectName or subjectAltName
      --extension) for the corresponding signing certificate
     reqPolicy                    PolicyInformation OPTIONAL,
     dcs                       GeneralName OPTIONAL,
     nonce                        Integer,
     reqTime                      ReqTime OPTIONAL,
     extensions                   Extensions OPTIONAL   }

ReqTime ::= CHOICE  {
     genTime                      GeneralizedTime,
     timeStampToken               TimeStampToken   }
```

In situations where the Data Certification Server will verify the identity of the requester (i.e., when the service field is cpd), the data certification request MUST be signed by the requester using the signerInfos field.

Similarly, in situations where the Data Certification Server will certify the time included in the request (i.e., when stipulated by the policy of the Data Certification Server), the data certification request MUST include the reqTime field in DCSReqInfo. Thus, when verifying a public key certificate, the presence of this field indicates the time for which the validity and revocation status of the certificate SHOULD be reported. If this field is not present, the current time is assumed. TimeStampToken is defined in Sect 2.4 of [TSA].

PolicyInformation is defined in Section 4.2.1.5 of [CCP]. The reqPolicy field SHOULD indicate the policy under which the certification is requested. This field MUST be checked by the DCS to verify agreement with its own policy. The absence of this field indicates that any policy is acceptable.

The Data type is defined to be either the message itself, a hash of the message (this allows a signature indicating possession of private data to be certified) or the certificate to be verified.

```
Data ::= CHOICE  {
     message                [0]  Message,
     messageimprint         [1]  MessageImprint,
     certs                  [2]  SEQUENCE SIZE (1..MAX) OF
                                      TargetandChain   }
```

In order to specify the format (i.e. the type) of the message so that it may be parsed and understood by the DCS or any verifying entity, we define the Message data type.

```
Message ::= SEQUENCE  {
     format                      MESSAGECLASS.&id,   --objid
     rawdata                     MESSAGECLASS.&Type  --open type
}
```



```
MESSAGECLASS ::= CLASS  {
     &id                         OBJECT IDENTIFIER UNIQUE,
     &Type                                                 }
WITH SYNTAX  { &Type IDENTIFIED BY &id }
```

Possible message types include id-data and id-signedData [CMS].

```
id-data OBJECT IDENTIFIER ::= { iso(1) member-body(2)
     us(840) rsadsi(113549) pkcs(1) pkcs7(7) 1 }

id-signedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
     us(840) rsadsi(113549) pkcs(1) pkcs7(7) 2 }
```

In particular, if the message type is id-signedData (or any other

message type that allows more than one signature) and more than one SignerInfo (or signature) is present under service type cs, the DCS MUST verify all signatures present. In this case the failure of any one signature to verify will result in the failure of the entire certification.

If the requester prefers to send a hash of the message instead, the MessageImprint data type SHOULD be used.

```
MessageImprint ::= SEQUENCE  {
     hashAlgorithm                AlgorithmIdentifier,
     hashedMessage                OCTET STRING  }
```

The hash algorithm indicated in the hashAlgorithm field SHOULD be a "strong" hash algorithm (that is, it SHOULD be one-way and collision resistant). It is up to the Data Certification Server to decide whether or not the given hash algorithm is sufficiently "strong" (based on the current state of knowledge in cryptanalysis and the current state of the art in computational resources, for example).

The hashedMessage field SHOULD contain the hash of the DER encoding of the message expressed as a Message data type. The hash is represented as an OCTET STRING.

```
TargetandChain ::= SEQUENCE {
     target                       Certificate,
     chain                        SEQUENCE SIZE (1..MAX) OF Certificate
                                        OPTIONAL,
     pathProcInput                PathProcInput OPTIONAL  }

PathProcInput ::= SEQUENCE {
     acceptablePolicySet          CertPolicySet,
     inhibitPolicyMapping         BOOLEAN DEFAULT FALSE,
     explicitPolicyReqd           BOOLEAN DEFAULT FALSE }
```

The certs field SHOULD contain the certificate to be certified. Each certificate SHALL be included in a separate instance of TargetandChain. The target field SHALL contain the cert to be verified and the chain field, if present, MUST indicate the chain of trust to be used when certifying the certificate. The pathProcInput field, if present, SHOULD indicate the acceptable policy set and initial settings for explicit-policy-indicator and inhibit-policy-mapping indicators to be

used in X.509 public key certificate path validation (see [CCP]).

Extensions are described in Section 4.2 of [CCP]. The criticality of the extensions MUST be honoured by conformant DCSs and clients (e.g. requests and responses containing critical extensions which are not understood MUST be rejected).

A data certification token is as follows. It is encapsulated as a SignedData construct [CMS]. The content is of type DCSInfo, which is indicated by the OID:

```
DCSInfo OBJECT IDENTIFIER  ::= { ??????  }
```

The data certification token MUST contain only the signature of the DCS.

```
DCSInfo ::= SEQUENCE  {
     dcsReqInfo                DCSReqInfo,
       --MUST be the same value as the dcsReqInfo field in DCSReqData
     messageImprint               MessageImprint,
       --if the data field in DCSReqData is MessageImprint, this
       --MUST contain that same value, otherwise it contains a hash of
       --the data field in DCSReqData using the hash algorithm
       --specified in the digestAlgorithm parameter of SignerInfo in
       --the data certification token
     reqSignature                 SignerInfo OPTIONAL,
       --MUST be present if service field of dcsReqInfo is cpd
       --MUST be the same value as the signerInfo field in the data
       --certification request
     policy                       PolicyInformation,
     status                       PKIStatusInfo,
     time                         DCSTime,
     chainCerts              [0]  SEQUENCE OF Certificate OPTIONAL,
       --if present, MUST indicate the chain of trust that was used by
       --the DCS to verify the signature or certificate in DCSReqData
     crls                    [2]  SEQUENCE OF CertificateList OPTIONAL,
     policyReturnInfo        [3]  SEQUENCE OF PolicyReturnInfo OPTIONAL,
     extensions                   Extensions OPTIONAL  }

DCSTime ::= CHOICE  {
     genTime                      GeneralizedTime,
     timeStampToken               TimeStampToken   }

PolicyReturnInfo ::= SEQUENCE {
     policies                     SEQUENCE OF PolicyInformation,
     mappings                     SEQUENCE OF PolicyMappingsSyntax }
```

PKIStatusInfo is defined in Section 3.2.3 of [CMP]. If the PKIStatus field has value `waiting' (3), then this token is a receipt, as defined in Section 2. Otherwise, the status field indicates whether or not the data certification request was fulfilled and, if not, failInfo indicates the reason it was rejected. A valid data certification token will have a PKIStatus field with value `granted' (0). For the purposes of the DCS, we define PKIFailureInfo for use in PKIStatusInfo.

```
PKIFailureInfo ::= BITSTRING  {
    badAlg           (0),
      -- unrecognized or unsupported Algorithm Identifier
    badMessageCheck  (1),
      -- integrity check failed (e.g., signature did not verify)
    badRequest       (2),
      -- transaction not permitted or supported
    badTime          (3),
      -- messageTime was not sufficiently close to the system time,
      -- as defined by local policy
    badCertId        (4),
      -- no certificate could be found matching the provided criteria
    badDataFormat    (5),
      -- the data submitted has the wrong format
    wrong   (6),
      -- the  indicated in the request is different from the
      -- one creating the response token
    incorrectData    (7),
      --the requester's data (i.e. signature) is incorrect
      --(i.e. invalid)
    missingTimeStamp (8),
      -- when the timestamp is missing but should be there (by policy)
    certInvalid      (9),
      -- the certificate fails to validate against Section 6 of [CCP]
    certRevoked      (10),
      -- the certificate is revoked
    certExpired      (11),
      -- the certificate has expired
    certOnHold       (12),
      -- the certificate has been operationally suspended
    certNotActive    (13)
      -- the certificate was not active at the given time
}
```

The statusString field of PKIStatusInfo can be used to include reason text such as "CA's public key revoked".

CertId is defined in Section 7.5 of [CRMF].

The crls field (if present) SHOULD contain a sequence of certificate revocation lists that is sufficient to verify the chain of trust indicated in the chainCerts field.

The policyReturnInfo field indicates the policies and mappings that were processed during X.509 public key certificate path validation. PolicyMappingsSyntax is defined in [CCP].

The reqSignature, chainCerts and crls fields are included as OPTIONAL. They SHOULD be present, when policy dictates, for use as supplementary evidence when resolving possible disputes. Dispute resolution would most likely be handled by one or more humans, in an off-line

environment, and is beyond the scope of this document.

6. Transports

There is no mandatory transport mechanism in this document. All

mechanisms are optional.

6.1. File Based Data Certification Server Protocol

A file containing a data certification message MUST contain only the DER encoding of one PKI message, i.e. there MUST be no extraneous header or trailer information in the file.

Such files can be used to transport data certification messages using for example, FTP.

6.2. Socket Based Data Certification Server Protocol

The socket based protocol for data certification messages is identical to that used in [CMP] Section 5.2 except that port 309 MUST be used.

6.3. Data Certification Server Protocol Using Email

This section specifies a means for conveying ASN.1-encoded messages for the protocol exchanges described in Section 4 via Internet mail.

A simple MIME object is specified as follows.

```
   Content-Type: application/dcs
   Content-Transfer-Encoding: base64

   <<the ASN.1 DER-encoded Data Certification Server message, base64-
encoded>>
```

This MIME object can be sent and received using MIME processing engines and provides a simple Internet mail transport for Data Certification Server messages.

6.4. Data Certification Server Protocol via HTTP

This subsection specifies a means for conveying ASN.1-encoded messages for the protocol exchanges described in Section 4 via the HyperText Transfer Protocol.

A simple MIME object is specified as follows.

```
   Content-Type: application/dcs

   <<the ASN.1 DER-encoded Data Certification Server message>>
```

This MIME object can be sent and received using common HTTP processing engines over WWW links and provides a simple browser-server transport for Data Certification Server messages.

7. Security Considerations

This entire document discusses security considerations.

When designing a data certification service, the following considerations have been identified that have an impact upon the validity or "trust" in the data certification token.

1. The enclosed public key certificate is revoked or the signer's key is compromised and the corresponding public key certificate is revoked before the Data Certification Server acts upon the request. The Data Certification Server is REQUIRED to validate appropriate information within the request before it constructs the data certification token. It is therefore mandated that the DCS have access to current information regarding public key certificate status before it creates the token. In this situation, the certification process would produce an error.
2. The enclosed public key certificate is revoked or the signer's key is compromised and the corresponding certificate is revoked after the Data Certification Server acts upon the request. This is not a concern to the DCS once the Data Certification Server has constructed the token, as long as the compromise date in the CRL is not before the time of certification. If it is, this situation would have to be handled by off-line, possibly human-aided, means specific to the situation at hand.
3. The Data Certification Server's private key is compromised and the corresponding certificate is revoked. In this case, any token signed by the Data Certification Server cannot be trusted. For this reason, it is imperative that the Data Certification Server's key be guarded with proper security and controls in order to minimize the possibility of compromise. Nevertheless, in case the private key does become compromised, an audit trail of all the tokens generated by the DCS SHOULD be kept as a means to help discriminate between genuine and false tokens.
4. The DCS signing key MUST be of a sufficient length to allow for a sufficiently long lifetime. Even if this is done, the key will have a finite lifetime. Thus, any token signed by the DCS SHOULD be time stamped (if authentic copies of old CRLs are available) or certified again (if they aren't) at a later date to renew the trust that exists in the DCS's signature. Data certification tokens could also be kept with an Evidence Recording Authority [ISONR] to maintain this trust.
5. When there is a reason to believe that the DCS can no longer be trusted, its certificate MUST be revoked and placed on the

appropriate CRL. Thus, at any future time the tokens signed with the corresponding key will not be trusted.

6. In certain circumstances, a DCS may not be able to produce a valid response to a request (for example, if it is unable to compute signatures for a period of time). In these situations the DCS MUST wait until it is again able to produce a valid response and then respond to the request. Under no circumstances shall a DCS produce an unsigned response to a request.
7. This protocol assumes that the CA has conducted a test for proof of possession for each user's signing private key. If this is not the case, or when additional assurances are required, the certificate of the requester (resp. DCS) SHALL be included in the encapsulation of the data certification request (resp. data certification token) as an authenticated attribute.

8. Patent Information

The following United States Patents related to data certification

servers (notaries), listed in chronological order, are known by the authors to exist at this time. This may not be an exhaustive list. Other patents may exist or be issued at any time. Implementers of this protocol SHOULD perform their own patent search and determine whether or not any encumberences exist on their implementation.

4,309,569 Method of Providing Digital Signatures
(issued) January 5, 1982
(inventor) Ralph C. Merkle
(assignee) The Board of Trustees of the Leland Stanford Junior University

5,001,752 Public/Key Date-Time Notary Facility
(issued) March 19, 1991
(inventor) Addison M. Fischer

5,022,080 Electronic Notary
(issued) June 4, 1991
(inventors) Robert T. Durst, Kevin D. Hunter

5,136,643 Public/Key Date-Time Notary Facility
(issued) August 4, 1992
(inventor) Addison M. Fischer
(Note: This is a continuation of patent # 5,001,752.)

5,136,646 Digital Document Time-Stamping with Catenate Certificate
(issued) August 4, 1992
(inventors) Stuart A. Haber, Wakefield S. Stornetta Jr.
(assignee) Bell Communications Research, Inc.,

5,136,647 Method for Secure Time-Stamping of Digital Documents
(issued) August 4, 1992
(inventors) Stuart A. Haber, Wakefield S. Stornetta Jr.
(assignee) Bell Communications Research, Inc.,

5,373,561 Method of Extending the Validity of a Cryptographic Certificate
(issued) December 13, 1994
(inventors) Stuart A. Haber, Wakefield S. Stornetta Jr.
(assignee) Bell Communications Research, Inc.,

5,422,95 Personal Date/Time Notary Device
(issued) June 6, 1995
(inventor) Addison M. Fischer

9. References

[TSA] C. Adams, P. Cain, D. Pinkas, R. Zuccherato, "Time Stamp Protocols," draft-adams-time-stamp-0X.txt, 1998 (work in progress).

[CMP] C. Adams, S. Farrell, "Internet Public Key Infrastructure, Certificate Management Protocols," draft-ietf-pkix-ipki3cmp-0X.txt, 1998 (work in progress).

[CRMF] C. Adams, "Internet Public Key Infrastructure, Certificate Request Message Format," draft-ietf-pkix-crmf-0X.txt, 1998 (work in progress).

[CCP] R. Housley, W. Ford, W. Polk, D. Solo, "Internet Public Key Infrastructure, X.509 Certificate and CRL Profile," draft-ietf-pkix-ipki-part1-0X.txt, 1998 (work in progress).

[CMS] R. Housley "Cryptographic Message Syntax", draft-ietf-smime-cms-02.txt, 1998 (work in progress).

[ISONR] ISO/IEC 10181-5: Security Frameworks in Open Systems. Non-Repudiation Framework.

[RFC2119] Key works for use in RFCs to Indicate Requirement Levels, **S. Bradner, RFC 2119, March 1997.**

10. Authors' Addresses

Carlisle Adams
Entrust Technologies
750 Heron Road
Ottawa, Ontario

Robert Zuccherato
Entrust Technologies
750 Heron Road
Ottawa, Ontario

K1V 1A7
CANADA
cadams@entrust.com

K1V 1A7
CANADA
robert.zuccherato@entrust.com

APPENDIX A - Storage of Data and Token

A data certification token is useless without the data to which it applies. For this reason tokens and their related data MUST be securely stored together. The change of a single bit in either the data or the token renders the entire certification process for that data meaningless. Storage of tokens and data in a secure (e.g., tamper proof) environment is strongly RECOMMENDED.

When data and data certification tokens are stored together, the following ASN.1 data type MAY be used.

```
DataAndToken ::= SEQUENCE  {
     message                     Message,
     dcsToken                 DCSInfo  }
```

Note that this object does not need to be signed, as the data certification token already verifies the integrity of the data in the

message. Any supplementary information whose integrity needs to be protected SHOULD be part of the message or token.

APPENDIX B - Extending the Life of a Signature

We present an example of a possible use of this data certification service. It produces a stand-alone token that can be used to extend the life of a signature. This example assumes that we have total trust in the Data Certification Server.

Signature algorithms and keys have a definite lifetime. Therefore, signatures have a definite lifetime. The Data Certification Server can be used to extend the lifetime of a signature.

In order to extend the lifetime of a signature in this way, the following technique MAY be used.

A) The signature needs to be certified.

1) The signed message is presented to the Data Certification Server in the data field of DCSReqInfo under service type cs and an appropriate policy.

2) The Data Certification Server verifies that the signature and verification key are valid at that time by checking expiry dates and status information, and returns a data certification token.

B) The certified signature MUST be verified.

1) The signature of the Data Certification Server in data certification token SHALL be verified using the Data Certification Server's valid verification key.

In this situation the signer's signing key (and therefore, its signature) is only valid until some specified time T1. The DCS's signing key (and therefore, its signature) is valid until some specified

time T2 that is (usually) after time T1. Without certification, the signer's signature would only be valid until time T1. With certification, the signer's signature remains valid until time T2, regardless of subsequent revocation or expiry at time T1.

If the signature of the DCS is valid, the trust we have in the DCS allows us to conclude that the original signature on the data was valid at the time included in the dcsInfo field of the data certification token.

APPENDIX C - Verifying the Status of a Public Key Certificate

We now present an example of how to produce a stand-alone token that can be used to confirm the revocation status of a public key certificate.

CRLs and ARLs are updated according to a schedule at regular intervals. For some purposes, the granularity provided by the CRLs and ARLs is not fine enough. Up-to-date revocation status may be needed before the next CRL or ARL update. Since the DCS MUST have access to current information regarding public key certificate status, it can be used to verify the revocation status of a certificate in this situation.

In order to produce such a token, the following technique MAY be used.

A) The public key certificate needs to be certified.

   1) The certificate is presented to the Data Certification Server in the data field of DCSReqInfo under service type cpkc and an appropriate policy.

   2) The Data Certification Server verifies that the public key certificate is valid and that it hasn't been revoked and then returns a data certification token.

B) The data certification token MUST be verified.

   1) The signature of the Data Certification Server in the data certification token SHALL be verified using the Data Certification Server's valid verification key.

This data certification token can now be used when verifying signatures using the key contained in the public key certificate. This service provided by the DCS can be thought of as a supplement to the usual method of checking revocation status.