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

Internet Draft
expires in six months
February 27, 1997

Notary Protocols

<draft-adams-notary-01.txt>

Abstract

This document describes a general notary service and the protocols to be used when communicating with it. The Notary Authority is a Trusted Third Party (TTP) that can be used as one component in building reliable non-repudiation services (see [ISONR]). Useful Notary Authority responsibilities in a PKI are to validate signatures and to provide up-to-date information regarding the status of certificates. We give examples of how to use the notary to extend the lifetime of a signature beyond key expiry or revocation and to query the notary regarding the status of a 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 Notary Authority (NA) is a Trusted Third Party that verifies the correctness of specific data submitted to it. The Notary Authority provides the notary 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 notarizing possession of data or another entity s signature, the NA 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 NA's CA, or the root CA in a hierarchy). The NA MAY be able to rely on all relevant CRLs and ARLs, or the NA MAY need to supplement this with access to more current status information from the CA. It then includes a trusted time and creates a notary token. (See Appendix B.)

When notarizing a certificate, the NA 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 NA MAY be able to rely on all relevant CRLs and ARLs, or the NA 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 Notary Token. (See Appendix C.)

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

In all cases, the trust that PKI entities have in the Notary Authority is transferred to the contents of the notary token (just as trust in a CA is transferred to the certificates that it issues). As a particular example, a notary 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 Notary Authority

The Notary Authority is MAY to:

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

certificates and produce a signed notary token attesting to the validity and revocation status of the certificate, if asked by the requester.

3. include a monotonically incrementing value of the time of day or a time stamp token into its notary token.
4. include within each signed notary token an identifier to uniquely determine the trust and validation policy used for this signature.
5. sign each notary token using a key generated exclusively for this purpose and have this property of the key indicated on the corresponding certificate.

6. indicate in the token whether or not the signature or certificate verified, and if not, the reason the verification failed.
7. provide a signed receipt (i.e., in the form of an appropriately defined notary token) to the requester, where appropriate, as defined by policy.

3. Notary Transactions

As the first transaction of this mechanism, the requesting entity requests a notarization by sending a request (which is or includes a notary request, as defined below), including the data for which validity and/or possession is to be notarized, to the Notary Authority. Upon receiving the request, the Notary Authority reviews and checks the validity of the request. A valid request is of the form described in Section 5 of this document and can be properly decoded. If the request is valid, the Notary Authority performs the notarization and sends a response (which is or includes a notary token, as defined below) to the requesting entity. Otherwise, the Notary Authority returns an error message (i.e., in the form of an appropriately defined notary 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 NA, the correct data imprint, a valid signature, and satisfactory status, service and policy fields. Since the NA 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 NA

The NA MUST sign all notary 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-notary.** This extension

MUST be critical.

```
id-kp-notary    OBJECT IDENTIFIER ::= {id-kp  ??}
  -- Notarizing 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 Notary Service is required.

```
ServiceType ::= INTEGER  { npd(1), ns(2), nc(3) }
```

The value npd (Notarize Possession of Data) is used when only the signature on the notary request (i.e., possession of the data in the request) is to be verified. In this case the Notary Authority 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 ns (Notarize 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 certificate revocation or expiry.

The value nc (Notarize Certificate) is used when the validity and revocation status of the certificate included in the request is 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 NA MAY support any subset of the above services.

Upon receiving a signed request for either service ns or nc the NA MUST also verify the signature on the request as is done for the npd service. Note however, that signed requests for the ns or nc service are not required.

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

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

The notary request MUST contain only the signature of the requester.

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

```
NotaryReqData ::= SEQUENCE  {
     notaryReqInfo                NotaryReqInfo,
     data                         Data
       --the data to be notarized
       --this field MUST be of type Message if the service type is ns
       --and of type SEQUENCE OF Certificate if the service type is nc
}
```

The notaryReqInfo field contains information pertaining to the notary request.

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

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

In situations where the Notary Authority will verify the identity of the requester (i.e., when the service field is npd), the notary request MUST be signed by the requester using the signerInfos field.

Similarly, in situations where the Notary Authority will certify the time included in the request (i.e., when stipulated by the policy of the Notary Authority), the notary request MUST include the reqTime field in NotaryReqInfo. Thus, when verifying a 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 notarization is requested or the policy for which certificate validity is to be reported. This field MUST be checked by the NA to verify agreement with its own policy or to determine certificate validity. 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 notarized) or the certificate to be verified.

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

In order to specify the format (i.e. the type) of the message so that it may be parsed and understood by the NA 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 }
```

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 Notary Authority 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.

The cert field SHOULD contain the certificate to be notarized. If the sequence has length greater than 1, then the certificates MUST indicate a chain of trust to be used when notarizing the certificate.

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

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

The notary token MUST contain only the signature of the NA.

```
NotaryInfo ::= SEQUENCE  {
     notaryReqInfo                NotaryReqInfo,
       --MUST be the same value as the notaryReqInfo field in
       --NotaryReqData
     messageImprint               MessageImprint,
       --if the data field in NotaryReqData is MessageImprint, this
       --MUST contain that same value, otherwise it contains a hash of
       --the data field in NotaryReqData using the hash algorithm
       --specified in the digestAlgorithm parameter of SignerInfo in
       --the notary token
     reqSignature                 SignerInfo OPTIONAL,
       --MUST be present if service field of notaryReqInfo is npd
       --MUST be the same value as the signerInfo field in notary
request
     policy                       PolicyInformation,
     status                       PKIStatusInfo,
     time                         NotaryTime,
     chainCerts              [0]  SEQUENCE OF Certificate OPTIONAL,
       --if present, MUST indicate the chain of trust that was used by
       --the NA to verify the signature or certificate in NotaryReqData
     crls                    [2]  SEQUENCE OF CertificateList OPTIONAL
   }

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

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 notary request was fulfilled and, if not, failInfo indicates the reason it was rejected. A valid notary token will have a PKIStatus field with value granted (0). For the purposes of the NA, 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
    wrongAuthority   (6),
      -- the authority 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 and authority revocation lists that is sufficient to verify the chain of trust indicated in the chainCerts field.

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

6.1. File Based Notary Protocol

A file containing a notary 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 notary messages using for example, FTP.

6.2. Socket Based Notary Protocol

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

6.3. Notary 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/notary
   Content-Transfer-Encoding: base64

   <<the ASN.1 DER-encoded Notary message, base64-encoded>>
```

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

6.4. Notary 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/notary

   <<the ASN.1 DER-encoded Notary 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 Notary messages.

7. Security Considerations

This entire document discusses security considerations.

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

1. **The enclosed certificate is revoked or the signer s key is compromised and the corresponding certificate is revoked before** the notary acts upon the request. The notary is MAY to validate appropriate information within the request before it constructs the notary token. It is therefore mandated that the NA have access to current information regarding certificate status before it creates the token. In this situation, the notarization process would produce an error.

2. The enclosed certificate is revoked or the signer s key is compromised and the corresponding certificate is revoked after the notary acts upon the request. This is not a concern to the NA once the notary has constructed the token, as long as the

compromise date in the CRL is not before the time of notarization. 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 notary s private key is compromised and the corresponding certificate is revoked. In this case, any token signed by the notary cannot be trusted. For this reason, it is imperative that the notary 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 NA SHOULD be kept as a means to help discriminate between genuine and false tokens.
4. The NA 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 NA SHOULD be time stamped (if authentic copies of old CRLs are available) or notarized again (if they aren t) at a later date to renew the trust that exists in the NA s signature. Notary 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 NA can no longer be trusted, the authority s certificate MUST be revoked and placed on the appropriate ARL. Thus, at any future time the tokens signed with the corresponding key will not be trusted.

6. **In certain circumstances, an NA 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 NA MUST wait until it is again able to produce a valid response and then respond to the request. Under no circumstances shall an NA 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 reqester (resp. NA) SHALL be included in the encapsulation of the notary request (resp. notary token) as an authenticated attribute.

8. References

[TSA] C. Adams, P. Cain, D. Pinkas, R. Zuccherato, Time Stamp Protocols, draft-adams-time-stamp-0X.txt, 1997 (work in progress).

[CMP] C. Adams, S. Farrell, Internet Public Key Infrastructure, Certificate Management Protocols, draft-ietf-pkix-ipki3cmp-0X.txt, 1997 (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, 1997 (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.**

9. Authors Addresses

Carlisle Adams
Entrust Technologies
750 Heron Road
Ottawa, Ontario
K1V 1A7
CANADA
cadams@entrust.com

Robert Zuccherato
Entrust Technologies
750 Heron Road
Ottawa, Ontario
K1V 1A7
CANADA
robert.zuccherato@entrust.com

APPENDIX A - Storage of Data and Token

A notary 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 notarization process for that data meaningless. Storage of tokens and data in a secure (e.g., tamper proof) environment is strongly RECOMMENDED.

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

```
DataAndToken ::= SEQUENCE  {
     message                        Message,
     notaryToken                    Content Info  }
```

Note that this object does not need to be signed, as the notary 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 notary 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 Notary Authority.

Signature algorithms and keys have a definite lifetime. Therefore, signatures have a definite lifetime. The Notary Authority 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 notarized.

1) The signed message is presented to the Notary in the data field of NotaryReqInfo under service type ns and an appropriate policy.

2) The Notary verifies that the signature and verification key are valid at that time by checking expiry dates and status information, and returns a notary token.

B) The notarized signature MUST be verified.

1) The signature of the Notary in notary token SHALL be verified using the Notary 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 NA s signing key (and therefore, its signature) is valid until some specified time T2 that is (usually) after time T1. Without notarization, the

signer s signature would only be valid until time T1. With notarization, the signer s signature remains valid until time T2, regardless of subsequent revocation or expiry at time T1.

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

APPENDIX C - Verifying the Status of a 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 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 NA MUST have access to current information regarding 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 certificate needs to be notarized.

1) The certificate is presented to the Notary in the data field of NotaryReqInfo under service type nc and an appropriate policy.

2) The Notary verifies that the certificate is valid and that it hasn t been revoked and then returns a notary token.

B) The notary token MUST be verified.

1) The signature of the Notary in notary token SHALL be verified using the Notary s valid verification key.

This notary token can now be used when verifying signatures using the key corresponding to the certificate. This service provided by the NA can be thought of as a supplement to the usual method of checking revocation status.