PKIX Working Group
Internet Draft
expires in six months

C. Adams(Entrust Technologies)
R. Zuccherato(Entrust Technologies)
July 29, 1997

Internet Public Key Infrastructure

Part VI: Notary Protocols

<draft-ietf-pkix-ipki6np-00.txt>

Status of this Memo

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]). We also give an example of how to use the notary to extend the lifetime of a signature beyond key expiry or revocation.

1. Introduction

A Notary Authority (NA) is a Trusted Third Party that verifies 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 and/or the validity and correctness of various types of data at a particular instant in time. When notarizing possession of data, 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) along with all relevant CRLs and

ARLs (Authority Revocation Lists). It then includes a trusted time and creates a notary token.

When notarizing data, the NA verifies the correctness of the data and creates a notary token. In this case, however, data "correctness" is not as focused in scope as signature correctness; the particular definition to be applied is therefore necessarily policy- and datatype-dependent. For example, the data may itself contain one or more signatures (where "correctness" relates to the validity of these signatures), or it may contain assertions (where "correctness" relates to the truth value of these statements), or it may contain a contract (where "correctness" relates to the legal validity of the document).

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 required to:

1. verify the correctness of the enclosed digital signature using all available and appropriate CRLs, ARLs, 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 correctness of the enclosed data with respect to explicitly stated policies using all available and appropriate resources and produce a signed notary token attesting to the validity of the data, 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 its signature.
5. indicate in the token whether or not the signature or data verified, and if not, the reason the verification failed.
6. 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 TimeStampReq, 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. If the request is valid, the Notary Authority performs the notarization and sends a response (which is or includes a TimeStampToken, as defined below) to the requesting entity. Otherwise, the Notary Authority returns an error message (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. The token can now be used to authenticate the correctness or possession of the data.

4. Request and Token Formats

The ServiceType type indicates which type of Notary Service is required.

```
ServiceType ::= INTEGER  { npd(1), nd(2), nb(3) }
```

The value npd (Notarize Possession of Data) is used when only the signature on the NotaryReq (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 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 nd (Notarize Data) is used when only the data included in NotaryReqInfo is to be verified. This verification may mean verifying a digital signature contained in the data, verifying the correctness of the data, or verifying the intent of parties to a contract contained in the data, for example. The exact interpretation of this service must be clearly indicated in the NA s policy statement, but is implementation and policy dependent, and thus beyond the scope of this document. The value nb (Notarize Both) is used when both the signature and data are to be verified. A given NA may support any subset of the above services.

A notary request is as follows.

```
NotaryReq ::= SEQUENCE  {
     notaryReqData               NotaryReqData,
     signature                   BIT STRING OPTIONAL
       --over the ASN.1 DER encoding of NotaryReqInfo, must be present
       --if the service field of NotaryReqInfo is npd or nb
}
```

The data and information that will be notarized is contained in the notaryReqData field.

```
NotaryReqData ::= SEQUENCE  {
     notaryReqInfo               NotaryReqInfo,
     data                        Data
       --the data to be notarized
       --this field must be of type Message if the service type is nd
       --or nb
}
```

The notaryReqInfo field contains information pertaining to the notary request.

```
NotaryReqInfo ::= SEQUENCE  {
     service                     ServiceType,
     requester                   GeneralName OPTIONAL,
      --must be present if the service field is npd or nb
```



```
     signatureAlgorithm          AlgorithmIdentifier,
      --must be present if the service field of NotaryReqInfo is
      --npd or nb
     certs                       SEQUENCE OF Certificate OPTIONAL,
     reqPolicy                   PolicyInformation OPTIONAL,
     notary                      GeneralName,
     reqTime                     TimeStampToken OPTIONAL   }
```

In situations where the Notary Authority will verify the identity of the requester (i.e., when the service field is npd or nb), the notary request must be signed by the requester using the signature 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. TimeStampToken is defined in Section 4 of PKIX Part 5 [PKIX5].

PolicyInformation is defined in Section 4.2.1.5 of PKIX Part 1 [PKIX1]. The reqPolicy field should indicate the policy under which the notarization is requested. This field must be checked by the NA to verify agreement with its own policy.

The Data type is defined to be either the message itself or a hash of the message. This allows a signature indicating possession of private data to be notarized.

```
Data ::= CHOICE  {
     message                   Message,
     messageimprint            MessageImprint  }
```

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 .

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.

A notary token is as follows.

```
NotaryToken ::= SEQUENCE  {
     notaryInfo                   NotaryInfo,
     signature                    BIT STRING,
      --over the ASN.1 DER encoding of NotaryInfo
}

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 signatureAlgorithm parameter
     signature                    BIT STRING OPTIONAL,
       --must be present if service field of notaryReqInfo is npd or nb
       --must be the same value as the signature field in NotaryReq
```

```
     policy                        PolicyInformation,
     status                        PKIStatusInfo,
     time                          NotaryTime,
     signatureAlgorithm            AlgorithmIdentifier,
     certId                        CertId,
       --must refer to the NA s public verification certificate
     certs                         SEQUENCE OF Certificate OPTIONAL,
       --if present, must indicate the chain of trust used to verify the
       --signature
     crls                          SEQUENCE OF CertificateList OPTIONAL
   }

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

PKIStatusInfo is defined in Section 3.2.3 of PKIX Part 3 [PKIX3]. If the PKIStatus field has value waiting (3), then this token is a receipt, as defined in Section 2. Otherwise, the status field is present to indicate whether or not the notary request was fulfilled and, if not, the reason it was rejected. A valid NotaryToken will have a PKIStatus field with value granted (0).

CertId is defined in Section 3.2.4 of PKIX Part 3 [PKIX3].

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 certs field.

The signature, certs 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. 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/x-pkix6
   Content-Transfer-Encoding: base64

   <<the ASN.1 DER-encoded PKIX-6 message, base64-encoded>>
```

This MIME object can be sent and received using MIME processing engines and provides a simple Internet mail transport for PKIX-6 messages.

7. 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 requester s key is compromised and the corresponding certificate is revoked before the notary acts upon the request. The notary is required 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. In this situation, the notarization would not occur.

2. The requester 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 again at a later date to renew the trust that exists in the NA s signature.

8. References

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

[PKIX1] R. Housley, W. Ford, W. Polk, D. Solo, Internet Public Key Infrastructure, Part I: X.509 Certificate and CRL Profile, draft-ietf-pkix-ipki-part1-0X.txt, 1997 (work in progress).

[PKIX3] C. Adams, S. Farrell, Internet Public Key Infrastructure, Part

III: Certificate Management Protocols, draft-ietf-pkix-ipki3cmp-0X.txt, 1997 (work in progress).

[PKIX5] C. Adams, P. Cain, D. Pinkas, R. Zuccherato, Internet Public Key Infrastructure, Part V: Time Stamp Protocols, draft-ietf-pkix-ipki5tsp-00.txt, 1997 (work in progress).

9. Authors Addresses

Carlisle Adams
Entrust Technologies
750 Heron Road, Suite 800
Ottawa, Ontario
K1V 1A7
CANADA
cadams@entrust.com

Robert Zuccherato
Entrust Technologies
750 Heron Road, Suite 800
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                 NotaryToken  }
```

Note that this object does not need to be signed, as the notary token already verifies the signature on 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 general 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 nd and an appropriate policy.

2) The Notary verifies that the signature and verification key are valid at that time by checking expiry dates, CRLs and ARLs, and returns a NotaryToken.

B) The notarized signature must be verified.

1) The signature of the Notary in NotaryToken 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 NotaryToken. Notice that in this example the validity of the original signature can be confirmed from the verification of one signature (the NA s) instead of two signatures (the Time Stamp Authority s and the original), but at the cost of putting more trust in the Trusted Third Party.