Traceable Anonymous Certificate
draft-ietf-pkix-tac-00.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on December 4, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2008).
Abstract

Public Key Infrastructure (PKI) provides a powerful means of authenticating individuals, organizations, and computers (e.g., web servers). However, when individuals use certificates to access resources on the public Internet, there are legitimate concerns about personal privacy, and thus there are increasing demands for privacy enhancing techniques on the Internet. In a PKI, an authorized entity such as a certification Authority (CA) or a Registration Authority (RA) may be perceived, from a privacy perspective, as a "big brother," even when a CA issues a certificate containing a Subject name that is a pseudonym. This is because such entities can always map a pseudonym in a certificate they issued to the name of the real user to whom it was issued. This document defines a practical architecture and protocols for offering privacy for a user who requests and uses an X.509 certificate containing a pseudonym, while still retaining the ability to map such a certificate to the real user who requested it. The architecture is compatible with IETF certificate request protocols such as PKCS10 [2] CRMF [3]. The architecture separates the authorities involved in issuing a certificate: one for verifying ownership of a private key (Anonymous Issuer) and the other for validating the contents of a certificate (Blind Issuer). The end-entity (EE) certificates issued under this model are called Traceable Anonymous Certificates (TACs).

Conventions used in this document

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 [1].

Table of Contents

1. Introduction............................................... 3
2. General Overview......................................... 4
3. Requirements............................................... 5
4. The Traceable Anonymous Certificate Model............... 6
5. Issuing a TAC............................................. 7
   5.1. Steps in Issuing a TAC................................ 9
   5.2. Mapping a TAC to a User’s Real Identity......... 12
1. Introduction

A Public Key Infrastructure (PKI) typically serves to identify the holder of a private key (to the corresponding public key in a certificate), in a standard fashion. The public key, identity, and related information, are signed by an entity acting as a Certification Authority (CA) as specified in X.509 and as profiled for use in the Internet [2]. During the past decade, PKIs have been widely deployed to support various types of communications and transactions over the Internet [6]. However, with regard to privacy on the Internet, a PKI is generally not supportive of privacy, at least in part because of the following issues:

- A certificate typically contains, in the Subject field the true identity of the user to whom it was issued. This identity is disclosed to a relying party (e.g., a web site or the recipient of an SMIME message [19]) whenever the certificate holder presents it in a security protocol that requires a user to present a certificate. In some protocols, e.g., TLS and S/MIME, a user’s certificate is sent via an unencrypted channel, prior to establishing a secure communication capability.

- A certificate often is published by the CA, for example in a directory system, which may be widely accessible.

- An anonymous (end entity) certificate [8, 12] is one that indicates that the holder’s true identity is not represented in the subject field. (Such a certificate might more accurately be called pseudonymous since an X.509 certificate must contain an identifier to comply with PKI format standards, and a CA must not issue multiple certificates with the same
Subject name to different entities. However we use the more common term "anonymous" throughout this document to refer to such certificates.) Issuance of anonymous certificates could enhance user privacy.

There is however, a need to balance privacy and accountability when issuing anonymous certificates. If a CA/RA is unable to map an anonymous certificate to the real user to whom it was issued, the user might abuse the anonymity afforded by the certificate, because there would be no recourse for relying parties.

A CA or RA generally would be able to map an anonymous certificate to the user to whom it was issued, to avoid such problems. To do so the CA/RA would initially identify the user and maintain a database that relates the user’s true identity to the pseudonym carried in the certificate’s Subject field. In a traditional PKI, there is a nominal separation of functions between a RA and a CA, but in practice these roles are often closely coordinated. Thus either the RA or CA could, in principle, unilaterally map an autonomous certificate to the real user identity.

The architecture, syntax, and protocol conventions described in this document allow anonymous certificates to be issued and used in existing PKIs in a way that provides a balance between privacy and a conditional ability to map an anonymous certificate to the individual to whom it was issued. The conditional traceability offered by this model assumes strong separation between the RA and CA roles, and employs technical means (threshold cryptography and "blinded" signatures), to enforce that separation. (A blinded signature is one in which the value being signed is not made visible to the signer, via cryptographic means. Additional details are provided later.) The technical measures require that both the RA and CA collaborate to disclose the identity associated with an anonymous certificate.

2. General Overview

This section defines the notion of a traceable anonymous certificate (briefly TAC or anonymous certificate in this document). It is distinguished from a conventional pseudonymous certificate [7, 10, 11] in that a TAC containing a pseudonym in the Subject field will be conditionally traceable (as defined in Section 1). Note
that it is not trivial to design a system that issues anonymous certificates, consistent with Internet PKI standards, when additional constraints are imposed, as illustrated by the following scenarios.

- If a CA issues an anonymous certificate without verifying a true identity, it is untraceable, which provides inadequate recourse if the user to whom the certificate was issued abuses the anonymity it provides. (Even without the ability to trace an anonymous certificate to the corresponding user, the certificate can always be revoked, but this may not be a sufficient response to abuse.)

- If a CA issues an anonymous certificate but verifies the real identity and maintains a record of that identity, the CA can link the pseudonym in the Subject field to the real identity, hence a potential "big brother" problem.

- If the CA issues a certificate with a certificate containing a user-selected Subject name, and does not verify the user’s identity, the certificate is effectively untraceable.

- If CA issues an anonymous certificate using a blind signature (see below), the CA cannot verify the contents of the certificate, making the certificate untraceable and essentially forgeable. (If a CA signs a certificate without examining its content, even after verifying a user’s identity, certificates issued by the CA are essentially forgeable.)

To address the issues described above, we extend the simple separation-of-authority concept already defined in the RA/CA PKI model. First we restate the requirements in a more precise and concise fashion, and introduce a basic model for achieving the goals from a more general perspective.

3. Requirements

This document describes a new separation-of-authority model and protocols for certificate issuance in a way that enables issuing traceable anonymous certificates, while maintaining compatibility with the standards used in existing PKIs. To do this, the following requirements must be satisfied.
The traceable anonymous certificate MUST be a syntactically valid X.509 certificate in which the Subject field contains a pseudonym.

There must be technical means to counter a claim by a malicious user who later denies having participated in the activities that resulted in issuing a TAC. Specifically, when a user is identified and requests issuance of a TAC, the mechanisms employed MUST ensure that the user to whom the TAC is issued is the one who requested the TAC (unless that user transfers the private key to another party, unknown to the RA/CA).

The traceability and revocation functions MUST support a bi-directional linkage between a user’s true identity and the pseudonym in a certificate issued to the user. Thus the solution MUST enable determining a true identity from the anonymous certificate and vice versa, upon agreement among the authorities who collaborated to issue the certificate.

4. The Traceable Anonymous Certificate Model

A TAC is issued by a pair of entities that operate in a split responsibility mode: a Blind Issuer (BI) and an Anonymous Issuer (AI). The pair appear as a single CA to the outside world, e.g., they are represented by a single CA certificate. The public key in the CA certificate is used to verify certificates issued by this CA in the normal fashion, i.e., a relying party processes a TAC just like any other EE certificates.

In this model the BI acts as a RA. It interacts with a user to verify the user’s "real" identity, just like a normal RA. The BI maintains a database that can be used to map a TAC to the user to whom it was issued, but only with the cooperation of the AI. This mapping will be initiated only if there is evidence that the user to whom the TAC was issued has abused the anonymity provided by the TAC.

The AI acts as a CA. It validates a certificate request submitted by the user, using a standard certificate request protocol such as PKCS10. The AI performs the functions common to a CA, including a private key proof of possession (PoP) check, a name uniqueness
check among all certificates issued by it, assignment of a serial number, etc. To effect issuance of the TAC, the AI interacts with the BI, over a secure channel, to jointly create the signature on the TAC, and sends the signed TAC to the user. The AI does this without learning the user’s real identity (either from the user or from the BI).

The result of this split functionality between the BI and the AI is that neither can unilaterally act to reveal the real user identity. The AI has knowledge of the certificate issued to the user, but no knowledge of the user’s real identity. The BI knows the user’s real identity, but has no knowledge of the certificate issued to that user. Only if the AI and BI collaborate can they map the TAC issued to a user to the real identity of that user.

This system is not perfect. For example, it assumes that the AI and BI collaborate to reveal a user’s real identity only under appropriate circumstances. The details of the procedural security means by which this assurance is achieved are outside the scope of this document. Nonetheless, there are security benefits to adopting this model described in this document, based on the technical approach used to enable separation of the BI and AI functions. For example, the BI and AI can be operated by different organizations in geographically separate facilities, and managed by different staff. As a result, one can have higher confidence in the anonymity offered to a user by the system, as opposed to a monolithic CA operating model that relies only on procedural security controls to ensure anonymity.

5. Issuing a TAC

The follow subsections describe the procedures and the protocols employed to issue a TAC.

To begin, BI and AI collaborate to generate a public key pair (that represents the CA as seen by relying parties) using a threshold signature scheme. Such schemes have been defined for RSA and for DSA. The details of how this is accomplished depend on the algorithm in question, and thus are not described here. The reader is referred to [16] and[17] where procedures for implementing RSA and DSA threshold signatures are described, respectively. Note that this split signing model for certificate issuance is an especially simple case of a threshold signature;
the private key used to sign a TAC is divided into exactly two shares, one held by the BI and one held by the AI. Both shares must be used, serially, to create a signature on a TAC. After the key pair for the (nominal) CA has been generated and the private key split between the BI and the AI, the public key is published, e.g., in a self-signed certificate that represents the TAC CA.

Another public key cryptographic function that is an essential part of this system is called "blind signing". To create a blind signature one party encrypts a value to be signed, e.g., a hash value of a certificate, and passes it to the signer. The signer digitally signs the encrypted value, and returns it to the first party. The first party inverts the encryption it applied with the random value in the first place, to yield a signature on the underlying data, e.g., a hash value. This technique enables the signer to digitally sign a message, without seeing the content of the message. This is the simplest approach to blind signing; it requires that the public key needed to invert the encryption not be available to the blind signer. Other blind signing techniques avoid the need for this restriction, but are more complex. The tricky part of a cryptographic blinding function is that is must be associative and commutative, with regard to a public key signature function. Let B be a blinding function, B-INV is its inverse, and S is a public key signature. The following relationship must hold: B-INV( S( B( X ) ) ) = B-INV( B( S( X ) ) ) = S( X ). RSA can be use to blind a value with random value and to sign a blinded value, because the modular exponentiation operation used by RSA for both signature and for encryption is associative and commutative. (Blinding is also defined for signature algorithms like DSA, but the explanation is more complex, since DSA does not natively encrypt data.)

The TAC issuance process described below requires an ability for the BI, the AI, and the user to employ secure communication channels between one another. Use of TLS [18] is one suitable means to establish such channels, although other options also are acceptable. To this end, this document assumes TLS as the default secure communication channel, and thus requires that the BI and the AI have X.509 certificates that represent them. These certificates are
independent of the certificate that represents the CA (formed by the BI and the AI) and may be either self-signed or issued by other CA(s).

5.1. Steps in Issuing a TAC

Figure 1 depicts the procedures for issuing a TAC. The lines represent steps in the issuance process, and the numbers refer to these steps.

Step 1. A user authenticates himself to BI. This may be effected via an in-person meeting or electronically. The same sorts of procedures that RAs use for normal certificate issuance are used here. Such procedures are not standardized, and thus they are not described here in detail. For purposes of the TAC architecture, we require the BI to establish a record in a database for the user, and to generate a (locally) unique identifier, called the UserKey, that will serve as a (database) key for the record. The UserKey value MUST NOT be generated in a fashion that permits any external entity (including the AI) to infer a user’s real identity from its value.

It is RECOMMENDED that the UserKey be a random or pseudo-random value. Whenever the BI passes a UserKey to an external party, or accepts the UserKey from an external party (e.g., the AI), the value is embedded in digitally signed CMS object called a Token. The signature on a Token is generated by the BI using a private key employed only for this purpose.
(The corresponding public key is not disclosed to any other entity, since only the BI needs to verify its signature on a Token.)

The following ASN.1 syntax represents the UserKey:

UserKey ::= OCTET STRING

Step 2. BI presents to the user a data structure called a Token. The Token must be conveyed to the user via a secure channel, e.g., in person or via a secure communication channel. The secure channel is required here to prevent a wiretapper from being able to acquire the Token. For example, if the user establishes a one-way authenticated TLS session to the BI in Step 1, this session could be used to pass the Token back to the user.

The Token serves two purposes. During TAC issuance, the Token is used to verify that a request to the AI has been submitted by a user who is registered with the BI (and thus there is a record in BI’s database with the real identity of the user). This is necessary to ensure that the TAC can later be traced to the user. If there is a request to reveal the real identity of a user, the AI will release the Token to the entity requesting that a TAC be traced, and that entity will pass the Token to the BI, to enable tracing the TAC.

The Token is a CMS SignedData object [5], signed by the BI. The content (encapContentInfo) is just the UserKey. The signature (SignatureValue) is generated using the BI’s private signature key, corresponding to the public key present in the BI’s certificate. (Note that this certificate is just a certificate suitable for use with TLS, and is NOT the split-key certificate used to verify a TAC.) The certificate(certificates) MUST be present. Appendix 1 provides the ASN.1 syntax for the Token, as a profiled CMS SignedData object.

Step 3. The user prepares a certificate request in a standard format, e.g., PKCS10 [3] or CRMF [4]. The Subject field of the certificate contains a pseudonym generated by the user. It is anticipated that the CA (BI + AI) may provide software for users to employ in constructing certificate requests. If so, then this software can generate a candidate Subject name to minimize the likelihood of a collision. If the user selects a candidate pseudonym without such support, the likelihood of a Subject name collision probably will be greater, increasing the
likelihood that the certificate request will be rejected or that the AI will have to generate a pseudonym for the user.

After constructing the certificate request, the user sends it, along with the Token from Step 2, to the AI, via a secure channel. This channel MUST be encrypted and one-way authenticated, i.e., the user MUST be able to verify that it is communicating with the AI, but the AI MUST NOT be able to verify the real identity of the user. Typical use of TLS for secure web site access satisfies this requirement. The certificate request, PKCS10 [3] or CRMF [4] carries the Token from Step 2.

The Token is carried as an attribute in a certificate request (CertificationRequestInfo.attributes) where the attrType MUST be id-kisa-tac below in PKCS10 format and the Token is set to attrValues(CertRequest.controls) where the attrType MUST be id-kisa-tac below.

Step 4: The AI, upon receipt of the certificate request containing a Token, verifies that the request is consistent with the processing defined for the request protocol (CRMF or PKCS10). If a Subject name is present, it verifies that the proposed pseudonym is unique. If the Subject field contains a Subject name already issued by the AI, the AI MUST either reject the certificate request, or substitute a pseudonym it generates, depending on the policy of the TAC CA. If the certificate request is acceptable, the AI assigns a serial number and constructs a tbsCertificate (i.e., the final form of the certificate payload, ready to be signed).

The AI then computes a hash over this data structure and blinds the hash value. (The AI blinds the hash value using a key from a public-key encryption pair where neither key is ever made public. The other key from this pair is used by the AI in Step 6 to "un-blind" the signed hash value.)

The AI sends the blinded certificate hash and the Token to the BI, via a two-way authenticated and encrypted channel. The two-way authentication and encryption is required to ensure that the AI is sending these values to the BI, to allow the BI to verify that the values were transmitted by the AI, and to prevent a wiretapper from acquiring the Token.

A TLS session in which both parties employ certificates to authenticate one another is the RECOMMENDED way to achieve this communication.

TokenandBlindHash ::= SEQUENCE {
    token          Token,
    blindHashValue OCTET STRING }
blindHashValue is the blinded hash value for the tbsCertificate.

**Step 5:** The BI receives the Token and blinded certificate hash via the secure channel described above. First the BI verifies the signature on the Token to ensure that it is a legitimate Token generated by the BI. Next, the BI checks its database to ensure that the UserKey value from the Token is present. This check is performed to ensure that the BI has authenticated the user and entered the user’s real identity into the BI’s database. This ensures that the certificate issued by the AI to this user will be traceable, if needed. The BI uses its share of the threshold private signature key to sign the blinded certificate hash, and returns the TokenandBlindHash object to the AI, via the secure channel used in Step 4. (The whole data structure is returned to the AI so that the AI can use the Token to match this response to the request it issued to the BI.)

**Step 6:** Upon receipt of the TokenandBlindHash, the AI matches the Token against its list of outstanding requests to the BI. The AI then "un-blinds" the blindHashValue, using the other key from the key pair employed Step 4. This reveals the partially-signed certificate hash. The AI then applies its part of the split private key to complete the signature of the certificate for the user. It records the certificate and the Token value in its database, to enable later tracing of the certificate to the real user identity, if needed.

**Step 7:** The AI transmits the completed certificate to the user, via the response message from the request protocol employed by the user in Step 3, i.e., either CRMF or PKCS10. The user may now employ the certificate with any PKI-enabled application or protocol that makes use of X.509 certificates (consistent with the key usage, and EKU values in the certificate).

### 5.2. Mapping a TAC to a User’s Real Identity

If a user to whom a TAC has been issued abuses the anonymity provided by the TAC, the TAC can be traced to the identity of that user. Mapping a TAC to a user’s real identity is a four step process, described below and illustrated in Figure 2.
Figure 2. Revealing a TAC User’s Real Identity

Step A: The AI verifies the assertion by an aggrieved party that a TAC user has abused the anonymity provided by his TAC. The procedures used by AI to verify that such abuse has occurred are outside the scope of this document. No protocol is defined here for the interaction between the aggrieved party and AI. The only technical requirement is that the TAC of the offending user be provided to the AI. If AI determines that there is sufficient evidence of abuse to trace the TAC to the user, the AI revokes the TAC, by listing its serial number on the next CRL issued by the AI. (If the AI uses OCSP [13] or SCVP [14] to convey the revocation status of TACs, an equivalent procedure is employed.) If it is later determined that the revocation was not warranted, a new TAC can be issued, to preserve the anonymity of the user in future transactions.

Step B: The AI searches its database, e.g., based on the serial number in the TAC, to locate the Token that was passed between the AI and BI during the issuance process (Steps 5 and 6 above). The AI passes this Token to the aggrieved party via an encrypted and two-way authenticated channel. Encryption is required to prevent disclosure of the Token, and two-way authentication is required to ensure that the aggrieved party and the AI know that they are communicating with each other. Two-way authenticated TLS is the RECOMMENDED means of implementing this channel, though other approaches are allowed.

Step C and D: The aggrieved party transits the Token to the BI, via an encrypted and two-way authenticated channel. The channel MUST be encrypted to prevent disclosure of the Token, and two-way authentication is required to ensure that the aggrieved party and the BI know that they are communicating with each other.
The BI verifies its signature on the Token, to verify that this is a Token generated by it and presumably released to the aggrieved party by the AI. Next the BI searches its database using the UserKey value extracted from the Token. The BI retrieves the user’s real identity and provides it to the aggrieved party. (By requiring the aggrieved party to interact with both the AI and the BI, the BI can verify that it is dealing with an aggrieved party, not with the AI acting unilaterally.)

6. Security Considerations

The anonymity provided by the architecture and protocols defined in this document is conditional. Moreover, if the user employs the same TAC for multiple transactions (with the same or different parties), the transactions can be linked through the use of the same TAC. Thus the anonymity guarantee is "weak" even though the user’s real identity is still hidden. To achieve stronger anonymity, a user may acquire multiple TACs, through distinct iterations of the protocol. Since each TAC is generated independently, it should not be possible for a relying party to discover a link between pseudonyms unless the tracing feature of this scheme is invoked.

This architecture uses the UserKey to link a TAC to the corresponding real user identity. The UserKey is generated in a fashion to ensure that it cannot be examined to determine a user’s real identity. UserKey values are maintained in two distinct databases: the BI database maps a UserKey to a real user identity, and the AI database maps a TAC to a UserKey. The UserKey is always carried in a signed data object, a Token. The Token is signed to allow the BI to verify its authenticity, to prevent attacks based on guessing UserKey values.

Threshold cryptography is employed to enable strong separation of the BI and AI functions, and to ensure that both must cooperate to issue certificates under the aegis of a TAC CA. Blind signatures are used with threshold cryptography to preserve the separation of functions, i.e., to prevent the BI from learning the hash value of the TAC issued by the AI.

Message exchanges between a user and the BI or the AI, between the AI and BI, and between an aggrieved party and the AI and BI
all make use of secure channels. These channels are encrypted to prevent disclosure of the Token value and of the pseudonym in the TAC request and response and in a tracing request. The channels are two-way authenticated to allow the AI and BI to verify their respective identities when communication with one another, and one-way authenticated to allow the user to verify their identities when he communicates with them. Two-way authentication is employed for communication between an aggrieved party and the AI and BI, to allow all parties to verify the identity of one another.

There is an opportunity for the AI to return the wrong UserKey to an aggrieved party, which will result in tracing a certificate to the wrong real user identity. This appears to be unavoidable in any scheme of this sort, since the database maintained by the BI is intentionally ignorant of any info relating a UserKey to a TAC.

7. IANA Considerations

This document does not require any IANA registration.

8. Acknowledgements

Tim Polk(NIST), Stefan Santesson(Microsoft), SeokLae Lee, JongHyun Baek, SoonTae Park(KISA), Taekyoung Kwon(Sejong Univ.), JungHee Cheon(Seoul National Univ.), YongDae Kim(Minnesota Univ.) have significantly contributed to work on the concept of TAC and identified security issues. Their comments enhance the maturity of the document.

9. References

9.1. Normative References


9.2. Informative References


[16] Shaohua Tang, "Simple Threshold RSA Signature Scheme Based on Simple Secret Sharing", 2005


DEFINITIONS IMPLICIT TAGS ::= 

BEGIN

-- EXPORTS All
-- The types and values defined in this module are exported for
-- use in the other ASN.1 modules. Other applications may use
-- them for their own purposes.

IMPORTS

-- Imports from RFC 3280 [PROFILE], Appendix A.1
AlgorithmIdentifier, Certificate, CertificateList,
CertificateSerialNumber, Name FROM PKIX1Explicit88
{ iso(1) identified-organization(3) dod(6)
  internet(1) security(5) mechanisms(5) pkix(7)
  mod(0) pkix1-explicit(18) }

-- Imports from CMS
SignedData FROM CryptographicMessageSyntax2004{ iso(1)
  member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
  smime(16) modules(0) cms-2004(24) }

UserKey ::= OCTET STRING

Token ::= SignedData

BlindCertificateHash ::= SEQUENCE {
  token Token,
  blindHashValue    OCTET STRING }

BlindSignature ::= OCTET STRING
id-KISA OBJECT IDENTIFIER ::= {iso(1) member-body(2) korea(410) kisa(200004)}

id-npki OBJECT IDENTIFIER ::= {id-KISA 10}

id-attribute OBJECT IDENTIFIER ::= {id-npki 1}

id-kisa-tac OBJECT IDENTIFIER ::= {id-attribute 1}

END
Author’s Addresses

SangHwan Park
Korea Information Security Agency
78, Garak-Dong, Songpa-Gu, Seoul, Korea
Email: shpark@kisa.or.kr

Haeryong Park
Korea Information Security Agency
78, Garak-Dong, Songpa-Gu, Seoul, Korea
Email: hrpark@kisa.or.kr

YooJae Won
Korea Information Security Agency
78, Garak-Dong, Songpa-Gu, Seoul, Korea
Email: yjwon@kisa.or.kr

JaeIl Lee
Korea Information Security Agency
78, Garak-Dong, Songpa-Gu, Seoul, Korea
Email: jilee@kisa.or.kr

Stephen Kent
BBN Technologies
10 Moulton Street Cambridge, MA 02138
Email: kent@bbn.com

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.
The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.