Network Working Group
Request for Comments: 5636
Category: Experimental

S. Park
H. Park
Y. Won
J. Lee
KISA
S. Kent
BBN Technologies

August 2009

Traceable Anonymous Certificate

Abstract

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 formats such as PKCS10 (RFC 2986) and CMC (RFC 5272). The architecture separates the authorities involved in issuing a certificate: one for verifying ownership of a private key (Blind Issuer) and the other for validating the contents of a certificate (Anonymity Issuer). The end entity (EE) certificates issued under this model are called Traceable Anonymous Certificates (TACs).

Status of This Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Table of Contents

1. Introduction

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.

A 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 CA as specified in X.509 [11] 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.

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 S/MIME message [18]) whenever the certificate holder presents it in a security protocol that requires a user to present a certificate. In some protocols, e.g., TLS, 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 that may be widely accessible.

- An anonymous (end entity) certificate [9] 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.

An anonymous certificate (Traceable Anonymous Certificate) in this document is issued by a pair of entities that operate in a split responsibility mode: a Blind Issuer (BI) and an Anonymity Issuer (AI). 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 facilitate 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 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.

1.1. 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].

2. General Overview

This section defines the notion of a Traceable Anonymous Certificate (referred to as TAC or anonymous certificate in this document). It is distinguished from a conventional pseudonymous certificate [8, 9] in that a TAC containing a pseudonym in the Subject field will be conditionally traceable (as defined 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 [12].

- 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 the 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 [16].

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 the 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, upon agreement among the authorities who collaborated to issue the certificate.

4. Traceable Anonymous Certificate Model

A TAC is an end entity (EE) certificate issued by a pair of entities that operate in a split responsibility mode: a Blind Issuer (BI) and an Anonymity 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 format 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. The details of how this is accomplished depend on the algorithm in question, and thus are not described here. The reader is referred to [15] where procedures for implementing RSA threshold signatures are described. A DSA-based threshold signature scheme will be incorporated into a future version of TAC [14].

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 used 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.

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 [17] 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).

Appendix B provides a top-level description of the application of TLS to these message exchanges.

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.

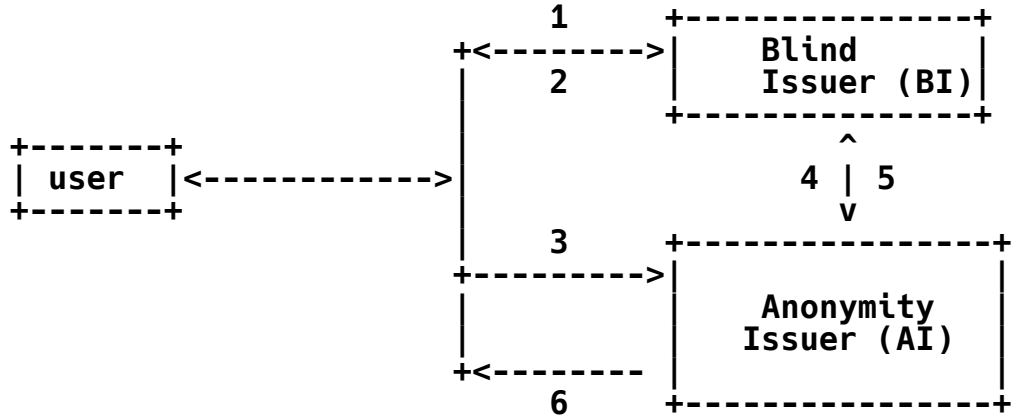

Figure 1. TAC Issuance Procedures

Step 1:

A user authenticates himself to the 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. (For example, if the user's name is used as an input to a one-way hash algorithm to generate the UserKey value, then additional random data must be used as an input to prevent simple guessing attacks.) Associated with the UserKey in this database is an expiration time. The expiration time is used by the BI and AI to reject session-level replay attacks in some exchanges, and to enable the BI and AI to garbage-collect database records if a user initiates but does not complete the certificate request process.

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 a digitally signed CMS object called a Token, accompanied by the timestamp noted above. The signature on a Token is generated by the BI. (Note that the certificate used is just a certificate suitable for use with CMS, and is NOT the split-key certificate used to verify TAC.)

The following ASN.1 syntax represents the UserKey and an expiration time:

```
UserKey ::= OCTET STRING
Timeout ::= GeneralizedTime
```

In the context of this specification, the GeneralizedTime value MUST be expressed in Greenwich Mean Time (Zulu) and MUST include seconds (YYYYMMDDHHMMSSZ).

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 the 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. If the BI does not perform its part of the certificate issuance procedure (in Step 6) before the Token expires, the BI can delete the Token from the database as a means of garbage collection. The timeout value in a Token is selected by the BI.

The Token is a ContentInfo with a contentType of id-kisa-tac-token and a content that holds a SignedData of CMS SignedData object [6], signed by the BI, where the eContent (EncapsulatedContentInfo) is a SEQUENCE consisting of the UserKey and Timeout, and eContentType MUST be id-data.

```
EncapsulatedContentInfo ::= SEQUENCE {
   eContentType ContentType, -- OBJECT IDENTIFIER : id-data
   eContent [0] EXPLICIT OCTET STRING OPTIONAL }
-- DER encoded with the input of 'SEQUENCE of the UserKey and
-- Timeout'

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

The signature (SignatureValue of SignerInfo) 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 (or certificates) MUST be present. Appendix A provides the ASN.1 syntax for the Token, as a profiled CMS ContentInfo object. Appendix C provides the CMS SignedData object profile for wrapping the Token.

```
   Token ::= ContentInfo
```

Upon receipt of the Token, the user SHOULD verify the signature using the BI public key and note the Timeout value to ensure that the certificate request process is completed prior to that time.

Step 3:

The user prepares a certificate request in a standard format, e.g., PKCS10 [3] or CMC [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 of PKCS10 [3] or CMC [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. The Token is set to attrValues (Certificate Request Controls) where the attrType MUST be id-kisa-tac in CMC [4] format. The TAC request message profile is described in the section 5.3.

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 format (PKCS10). If a Subject name is present, it verifies that the proposed pseudonym is unique. The AI also verifies the signature on the Token and, if it is valid, checks the Timeout value to reject a replay attack based on a "timed-out" Token.

A Token with an old Timeout value is rejected out-of-hand by the AI. (After a Token's Timeout time is reached, the AI deletes the Token from its cache.) Next, the AI compares the received Token against a cache of recent (i.e., not "timed out"), validated Tokens. The AI matches the resubmitted request to the original request, and responds accordingly. For example, if a duplicate is detected, the certificate request can be rejected as a replay.

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 CMS ContentInfo object of TokenandBlindHash 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.

The TokenandBlindHash is a CMS ContentInfo with a contentType of id-kisa-tac-tokenandblindhash and a content that holds a SignedData of CMS SignedData object [6], signed by the AI, where the eContent (EncapsulatedContentInfo) is a SEQUENCE consisting of the Token and BlindedCertificateHash, and eContentType MUST be id-data.

```
EncapsulatedContentInfo ::= SEQUENCE {
   eContentType ContentType, -- OBJECT IDENTIFIER : id-data
   eContent [0] EXPLICIT OCTET STRING OPTIONAL }
-- DER encoded with the input of 'SEQUENCE of the Token and
-- BlindedCertificateHash'
```

The signature (SignatureValue of SignerInfo) is generated using the AI's private signature key, corresponding to the public key present in the AI's certificate. (Note that this certificate is just a certificate suitable for use with TLS, and is NOT the split-key certificate used to issue a TAC.) The certificate (or certificates) MUST be present.

The following ASN.1 syntax represents the Token and BlindedCertificateHash:

```
   Token ::= ContentInfo
   BlinedCertificateHash ::= OCTET STRING
```

Token is the value of ContentInfo in the certificate request message (CertificationRequestInfo.attributes) from Step 3.

BlindedCertificateHash is the blinded hash value for the tbsCertificate.

Appendix A provides the ASN.1 syntax for the Token, as a profiled CMS ContentInfo object. Appendix C provides the CMS SignedData object profile for wrapping the Token.

```
TokenandBlindHash ::= ContentInfo
```

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 TokenandBlindHash generated by AI and then 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 and that the Token has not been used to authorize issuance of a certificate previously.

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. Each Token authorizes issuance of only one certificate, so the check also ensures that the same Token has not been used to authorize issuance of more than one certificate. These checks ensure 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 CMS SignedData back to the AI. The eContent of the SignedData is a SEQUENCE consisting of the Token and PartiallySignedCertificateHash.

The following ASN.1 syntax represents the Token and PartiallySignedCertificateHash:

```
Token ::= ContentInfo
PartiallySignedCertificateHash ::= OCTET STRING
```

Token is the token value of the TokenandBlindHash (where the eContent is a SEQUENCE consisting of the Token and PartiallySignedCertificateHash) from Step 4.

PartiallySignedCertificateHash is the signature value generated by BI's share of the threshold private signature key on BlindedCertificateHash from Step 4.

The TokenandPartiallySignedCertificateHash is a CMS ContentInfo with a contentType of id-kisa-tac-tokenandpartially and a content that holds a SignedData of CMS SignedData object [6], signed by the BI, where the eContent (EncapsulatedContentInfo) is a SEQUENCE consisting of the Token and PartiallySignedCertificateHash, and eContentType MUST be id-data.

```
EncapsulatedContentInfo ::= SEQUENCE {
   eContentType ContentType, -- OBJECT IDENTIFIER : id-data
   eContent [0] EXPLICIT OCTET STRING OPTIONAL }
-- DER encoded with the input of 'SEQUENCE of the Token and
-- PartiallySignedCertificateHash'
```

The signature (SignatureValue of SignerInfo) 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 issue a TAC.) The certificate (or certificates) MUST be present. Appendix A provides the ASN.1 syntax for the Token, as a profiled CMS SignedData object. Appendix C provides the CMS SignedData object profile for wrapping the Token.

```
   TokenandPartiallySignedCertificateHash ::= ContentInfo
```

Step 6:

Upon receipt of the TokenandPartiallySignedCertificateHash, the AI verifies the signature on the PartiallySignedCertificateHash, generated by BI and then 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 in 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. The AI transmits the completed certificate to the user, via the response message from the request protocol employed by the user in Step 3, 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 Extended Key Usage (EKU) values in the certificate). Note that the user should be prepared to accommodate delays in the certificate issuance process. For example, a connection between the user and the AI might fail sometime after the user submits a certificate request at the end of Step 3 and before the AI returns the certificate at the end of Step 6. If this happens, the user should resubmit the request. The AI and BI retain sufficient state to be able to match the resubmitted request to the original request, and respond accordingly. If the process failed in steps 5 or 6, the AI returns an error indication to the user.

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.

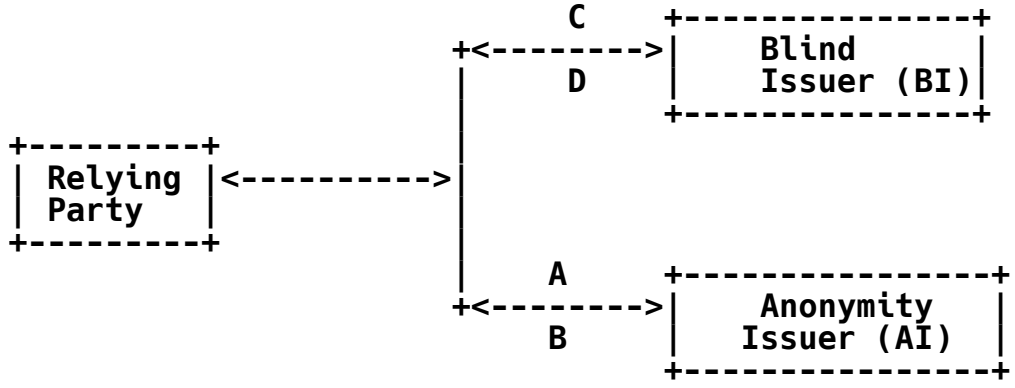

Figure 2. Revealing a TAC User's Real Identity

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 the 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 Certificate Revocation List (CRL) issued by the AI.

An AI unilaterally manages the CRL for a TAC. Because RFC 5280 implementations are not required to process indirect CRLs, we create a second certificate for the CA, under the TAC CA. Revoked EE certificates issued by the TAC CA are recorded on this CRL and validated using this second CA certificate.

This CA certificate will have the cRLSign bit set in the KeyUsage extension, but not the keyCertSign bit. The private key for this certificate will be held by the AI, so that it can issue CRLs unilaterally.

The Subject DN (Distinguished Name) will be the same in both CA certificates, which reinforces the notion that the CRL issuer is the same entity as the TAC issuer, and that this CRL is not an indirect CRL. Because the CRL issuer does not issue any certificates itself, there is no possible serial number conflict. This will be the only CA certificate issued under the TAC CA certificate (and thus it will be signed jointly by the BI and AI). We recommend that the CRL for this CA certificate be similarly long-lived, as it too needs to be signed by the BI and AI. Each EE TAC certificate MUST contain a CRL Distribution Point that points to the CRL issued by this CA, to ensure that relying parties know to check this CRL vs. the CRL that covers only the CRL CA. (If the AI uses the Online Certificate Status Protocol (OCSP) [13] 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.

Steps 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. If specified by the Certificate Policy (CP) for the TAC CA, the BI will independently determine that there is sufficient evidence of abuse to trace the TAC to the user, before proceeding. 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.)

5.3. TAC Request Message Format Profile

TAC request MAY use either PKCS10 or CMC. An AI MUST support PKCS10 and MAY support CMC.

5.3.1. PKCS10 Profile

This profile refines the specification in PKCS10 [3], as it relates to TAC. A Certificate Request Message object, formatted according to PKCS10, is passed to the AI.

This profile applies the following additional constraints to fields that may appear in a CertificationRequestInfo:

Version
: This field is mandatory and MUST have the value 0.

Subject
: This field MUST be present. If the value of this field is empty, the AI will generate a subject name that is unique in the context of certificates issued by this issuer. 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.

SubjectPublicKeyInfo
: This field specifies the subject's public key and the algorithm with which the key is used.

Attributes
: PKCS10 [3] defines the attributes field as key-value pairs where the key is an OID and the value's structure depends on the key. The attribute field MUST include the id-kisa-tac attribute, which holds the Token and is defined below. The

Attributes field MAY also contain X509v3 Certificate Extensions and any PKCS9 [7] extensionRequest attributes that the subscriber would like to have included in the certificate. The profile for extensions in certificate requests is specified in RFC 5280 [2].

5.3.2. CMC Profile

This profile refines the Certificate Request messages in Certificate Management over CMS in CMC [4], as they relate to TACs.

A Certificate Request message, formatted according to CMC [4], is passed to the AI.

With the exception of the public-key-related fields, the CA is permitted to alter any requested field when issuing a corresponding certificate.

This profile recommends the full PKI Request of the two types of PKI requests (Simple or Full PKI Request), and the PKI Request SHOULD be encapsulated in SignedData with an eContentType of id-cct-PKIData.

This profile applies the following additional constraints to fields that may appear in a Certificate Request Template of Certificate Request Message Format (CRMF) [5]:

Version
: This field MAY be absent, or MAY specify the request of a Version 3 Certificate. It SHOULD be omitted.

SerialNumber
: As per CRMF [5], this field is assigned by the CA and MUST be omitted in this profile.

SigningAlgorithm
: As per CRMF [5], this field is assigned by the CA and MUST be omitted in this profile.

Issuer
: This field is assigned by the CA and MUST be omitted in this profile.

Validity
: This field MAY be omitted. If omitted, the AI will issue a Certificate with Validity dates as determined by the TAC CA policy. If specified, then the CA MAY override the requested values with dates as determined by the TAC CA policy.

Subject
This field MUST be present. If the value of this field is empty, the AI MUST generate a subject name that is unique in the context of certificates issued by this issuer. 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.

PublicKey
This field MUST be present.

This profile also refines constraints that may appear in a Certificate Request controls: The Token is set to attrValues (in CertRequest.controls) where the attrType MUST be id-kisa-tac.

See Section 5.3.1, "PKCS10 Profile", for the certification request formats based on PKCS10.

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. If the TAC has a long validity interval, this increases the probability that the identity of a TAC user will be discovered, e.g., as a result of linking user transactions across multiple servers. Thus, we recommend that each TAC CA consider carefully how long the validity for a TAC certificate should be. In the course of issuing a TAC, the AI and the user interact directly. Thus, the AI may have access to lower-layer information (e.g., an IP address) that might reveal the user's identity. A user concerned about this sort of possible identity compromise should use appropriate measures to conceal such information, e.g., a network anonymity service such as Tor [10].

This document makes no provisions for certificate renewal or rekey; we recommend TAC users acquire new TACs periodically, to further reduce the likelihood of linkage. It also may be possible to determine the identity of a user via information carried by lower-

level protocols, or by other, application-specific means. For example, the IP address of the user might be used to identify him. For this reason, we recommend that a TAC be used primarily to access web services with anonymity. Note that the TAC architecture described in this document is not capable of using certificates for use with S/MIME, because there is no provision to issue two certificates (one for encryption and one for signatures) that contain the same (anonymous) Subject name. An analogous problem might arise if a user visits a site (and does not conceal his identity), the site deposits a "cookie" into the user's browser cache, and the user later visits a site and employs a TAC with the presumption of anonymity.

The use of a TAC is a tool to help a user preserve anonymity, but it is not, per se, a guarantee of anonymity. We recommend that each TAC CA issue certificates with only one lifetime, in order to avoid the complexity that might arise otherwise. If a TAC CA offered certificates with different lifetimes, then it would need to communicate this information from the BI to AI in a way that does not unduly compromise the anonymity of the user.

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. The Token also carries a Timeout value to allow the AI and BI to reject session-level replay attacks, and to facilitate garbage collection of AI and BI databases.

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. (The AI and BI must ensure that the threshold cryptographic scheme they employ does not provide an advantage to either party based on the way the key-splitting is effected.) 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.

A TAC CA MUST describe in its CP how long it will retain the data about certificates it issued, beyond the lifetime of these certificates. This will help a prospective TAC subject gauge the likelihood of unauthorized use of his identity as a result of a compromise of this retained data. It also alerts relying parties of the timeframe (after expiration of a certificate) in which an alleged abuse must be brought to the attention of the AI and BI, before the data linking a certificate to the real user identity is destroyed.

7. Acknowledgments

Tim Polk (NIST), Stefan Santesson (ACC-sec.com), Jim Schaad (Soaring Hawk), David A. Cooper (NIST), SeokLae Lee, JongHyun Baek, SoonTae Park (KISA), Taekyoung Kwon (Sejong University), JungHee Cheon (Seoul National University), and YongDae Kim (Minnesota University) have significantly contributed to work on the concept of TAC and have identified security issues. Their comments enhanced the maturity of the document.

8. References

8.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[3] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, November 2000.

[4] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, June 2008.

[5] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, September 2005.

[6] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3852, July 2004.

[7] Nystrom, M. and B. Kaliski, "PKCS #9: Selected Object Classes and Attribute Types Version 2.0", RFC 2985, November 2000.

8.2. Informative References

[8] S. Brands, "Rethinking public key infrastructures and digital certificates - Building in Privacy", PhD thesis, Eindhoven Institute of Technology, Eindhoven, The Netherlands, 1999.

[9] D. Chaum, "Blind signature system", CRYPTO '83, Plenum Press, page 153, 1984.

[10] "Tor: anonymity online", http://www.torproject.org.

[11] X.509, "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks", ITU-T Recommendation X.509, March 2000. Also available as ISO/IEC 9594-8, 2001.

[12] S. Rafaeli, M. Rennhard, L. Mathy, B. Plattner, and D. Hutchison, "An Architecture for Pseudonymous e-Commerce", AISB'01 Symposium on Information Agents for Electronic Commerce, pp. 33-41, 2001.

[13] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.

[14] Philip MacKenzie and Michael K. Reiter, "Two-Party Generation of DSA Signature", Crypto 2001.

[15] Shaohua Tang, "Simple Threshold RSA Signature Scheme Based on Simple Secret Sharing", in "Computational Intelligence and Security", CIS 2005, Part II, Springer, pp. 186-191, 2005.

[16] Taekyoung Kwon, Jung Hee Cheon, Yongdae Kim, Jae-Il Lee, "Privacy Protection in PKIs: A Separation-of-Authority Approach", in "Information Security Applications", WISA 2006, Springer, pp. 297-311, 2007.

[17] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[18] Ramsdell, B., Ed., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Certificate Handling", RFC 3850, July 2004.

Appendix A. Traceable Anonymous Certificate ASN.1 Modules

```
DEFINITIONS IMPLICIT TAGS ::=

--
--   Copyright (c) 2009 IETF Trust and the persons identified as
--   authors of the code.  All rights reserved.
--
--   Redistribution and use in source and binary forms, with or
--   without modification, are permitted provided that the following
--   conditions are met:
--
--   - Redistributions of source code must retain the above
--     copyright notice, this list of conditions and the following
--     disclaimer.
--
--   - Redistributions in binary form must reproduce the above
--     copyright notice, this list of conditions and the following
--     disclaimer in the documentation and/or other materials provided
--     with the distribution.
--
--   - Neither the name of Internet Society, IETF or IETF Trust, nor
--     the names of specific contributors, may be used to endorse or
--     promote products derived from this software without specific
--     prior written permission.
--
--
--
--   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
--   CONTRIBUTORS 'AS IS' AND ANY EXPRESS OR IMPLIED WARRANTIES,
--   INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
--   MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
--   DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS
--   BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
--   EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
--   TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
--   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
--   ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
--   OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
--   OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
--   POSSIBILITY OF SUCH DAMAGE.
--
--   This version of the ASN.1 module is part of RFC 5636;
--   see the RFC itself for full legal notices.
--
```

```
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
            ContentInfo, 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

Timeout ::= GeneralizedTime

BlinedCertificateHash ::= OCTET STRING

PartiallySignedCertificateHash ::= OCTET STRING

EncapsulatedContentInfo ::= SEQUENCE {
       eContentType ContentType, -- OBJECT IDENTIFIER : id-data
       eContent [0] EXPLICIT OCTET STRING OPTIONAL }

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

Token ::= ContentInfo

TokenandBlindHash ::= ContentInfo

TokenandPartiallySignedCertificateHash ::= ContentInfo

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}

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

id-kisa-tac-tokenandblindbash OBJECT IDENTIFIER ::= { id-kisa-tac 2}

id-kisa-tac-tokenandpartially OBJECT IDENTIFIER ::= { id-kisa-tac 3}

END
```

Appendix B. TAC Message Exchanges over Transport Layer Security

TAC 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 to prevent disclosure of the Token value and of the pseudonym in the TAC request and response and in a tracing request. The Transport Layer Security Protocol v1.2 (TLS) [17] is a suitable security protocol to protect these message exchanges, and this document recommends use of TLS to protect these exchanges. The following text describes how the handshake part of TLS should be employed to protect each type of exchange. Note that no specific cipher suites are specified for use here; the choice of suites is up to the client and servers, as is commonly the case.

B.1. Message Exchanges between a User and the BI or the AI

The channels between a User and the BI or the AI are one-way authenticated to allow the user to verify their identities when he communicates with them.

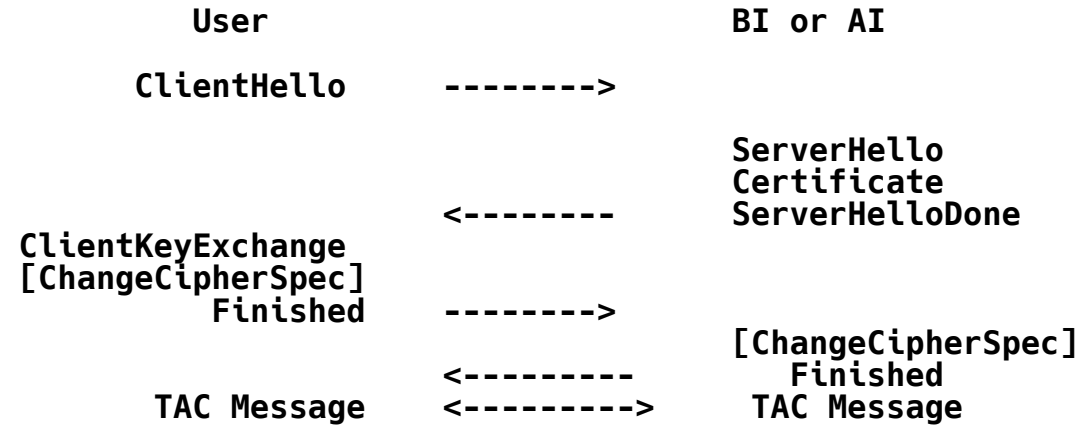

Figure 3. TAC Message exchanges between a User and the BI or the AI

B.2. Message Exchanges between the BI and the AI

The channels between the BI and the AI are two-way authenticated to allow the AI and BI to verify their respective identities when communication with one another.

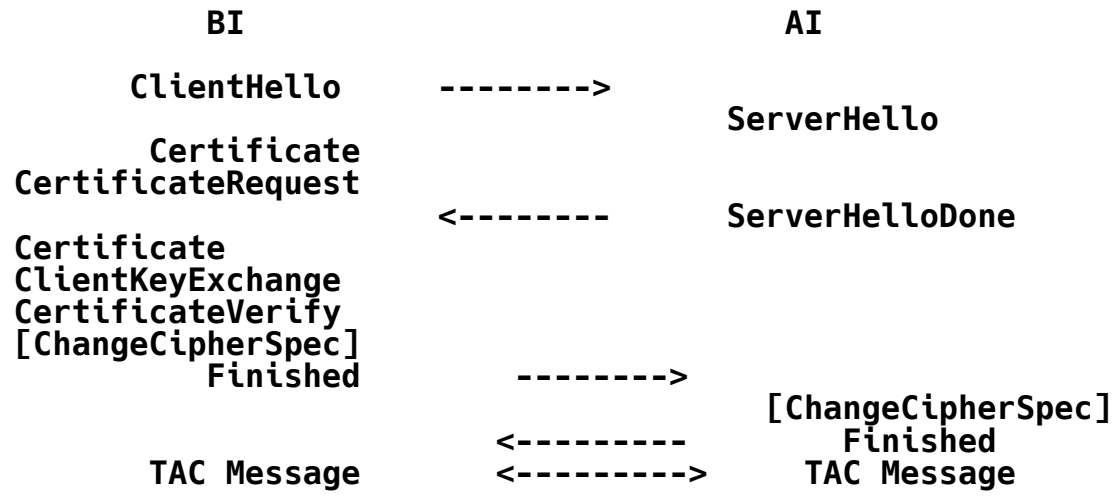

Figure 4. TAC Message exchanges between BI and AI

B.3. Message Exchanges between the Aggrieved Party and the AI or the BI

The channels between a User and the BI or the AI are two-way authenticated, to allow both parties to verify the identity of one another.

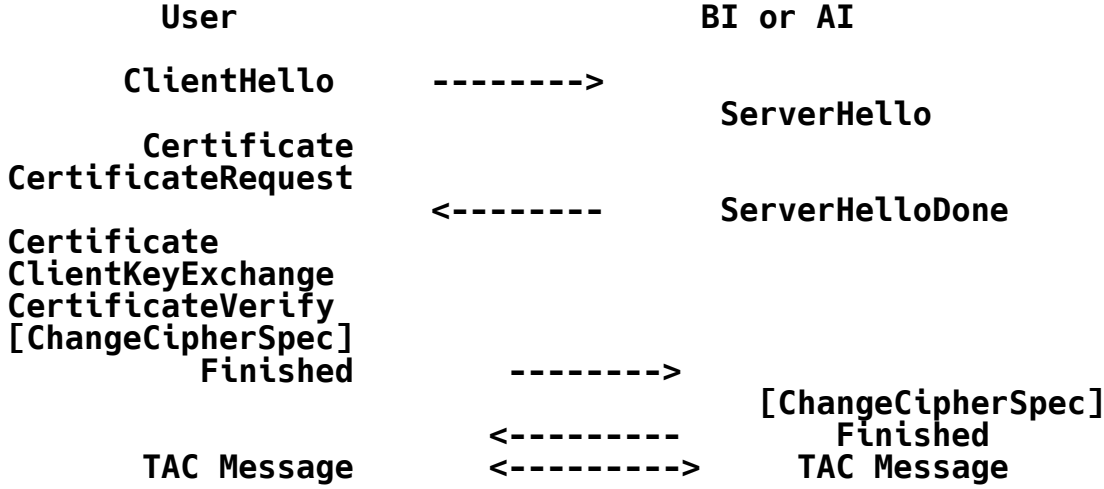

Figure 5. TAC Message Exchanges between an Aggrieved Party and the BI or the AI

Appendix C. Cryptographic Message Syntax Profile for TAC Token

Using the Cryptographic Message Syntax(CMS)[6], TAC Token is a type of signed-data object. The general format of a CMS object is:

```
ContentInfo ::= SEQUENCE {
            contentType ContentType,
            content [0] EXPLICIT ANY DEFINED BY contentType }

          ContentType ::= OBJECT IDENTIFIER
```

As a TAC is a signed-data object, it uses the corresponding OID, 1.2.840.113549.1.1.2.

C.1. Signed-Data Content Type

According to the CMS specification, the signed-data content type shall have ASN.1 type SignedData:

```
   SignedData ::= SEQUENCE {
           version CMSVersion,
           digestAlgorithms DigestAlgorithmIdentifiers,
           encapContentInfo EncapsulatedContentInfo,
           certificates [0] IMPLICIT CertificateSet OPTIONAL,
           crls [1] IMPLICIT RevocationInfoChoices OPTIONAL,
           signerInfos SignerInfos }

   DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier

   SignerInfos ::= SET OF SignerInfo
```

The elements of the signed-data content type are as follows:

Version
The version is the syntax version number. It MUST be 3, corresponding to the signerInfo structure having version number 3.

digestAlgorithms
This field specifies digest Algorithms.

encapContentInfo
This element is defined in Appendix C.1.1.

certificates
The certificates element MUST be included and MUST contain only the single PKI EE certificate needed to validate this CMS Object. The CertificateSet type is defined in section 10 of RFC3852 [6].

crls
The crls element MUST be omitted.

signerInfos
This element is defined in Appendix C.1.2.

C.1.1. encapContentInfo

encapContentInfo is the signed content, consisting of a content type identifier and the content itself.

```
EncapsulatedContentInfo ::= SEQUENCE{
    eContentType ContentType,
     eContent [0] EXPLICIT OCTET STRING OPTIONAL }

ContentType ::= OBJECT IDENTIFIER
```

The elements of this signed content type are as follows:

eContentType
The ContentType for an TAC Token is id-data and has the numerical value of 1.2.840.113549.1.7.1.

eContent
The content of a TAC Token is the DER-encoded SEQUENCE of UserKey and Timeout.

C.1.2. signerInfos

SignerInfo is defined under CMS as:

```
SignerInfo ::= SEQUENCE {
     version CMSVersion,
     sid SignerIdentifier,
     digestAlgorithm DigestAlgorithmIdentifier,
     signedAttrs [0] IMPLICIT SignedAttributes OPTIONAL,
     signatureAlgorithm SignatureAlgorithmIdentifier,
     signature SignatureValue,
     unsignedAttrs [1] IMPLICIT UnsignedAttributes OPTIONAL }
```

The contents of the SignerInfo element are as follows:

Version
The version number MUST be 3, corresponding with the choice of SubjectKeyIdentifier for the sid.

sid
The sid is defined as:

```
SignerIdentifier ::= CHOICE {
issuerAndSerialNumber IssuerAndSerialNumber,
subjectKeyIdentifier [0] SubjectKeyIdentifier }
```

For a TAC Token, the sid MUST be a SubjectKeyIdentifier.

digestAlgorithm
This field specifies digest Algorithms.

signedAttrs
The signedAttr element MUST be omitted.

SignatureAlgorithm
This field specifies the signature Algorithm.

Signature
The signature value is defined as:

```
SignatureValue ::= OCTET STRING
```

The signature characteristics are defined by the digest and signature algorithms.

UnsignedAttrs
unsignedAttrs MUST be omitted.

Authors' Addresses

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

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

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

JaeIl Lee
Korea Internet & 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