Workgroup: LAMPS
Internet-Draft:
draft-ounsworth-pq-composite-sigs-06
Published: 8 February 2022
Intended Status: Standards Track
Expires: 12 August 2022
Authors: M. Ounsworth M. Pala
Entrust CableLabs

Composite Signatures For Use In Internet PKI

Abstract

With the widespread adoption of post-quantum cryptography will come the need for an entity to possess multiple public keys on different cryptographic algorithms. Since the trustworthiness of individual post-quantum algorithms is at question, a multi-key cryptographic operation will need to be performed in such a way that breaking it requires breaking each of the component algorithms individually. This requires defining new structures for holding composite signature data.

This document defines the structures CompositeSignatureValue, and CompositeParams, which are sequences of the respective structure for each component algorithm. This document also defines processes for generating and verifying composite signatures. This document makes no assumptions about what the component algorithms are, provided that their algorithm identifiers and signature generation and verification processes are defined.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on 12 August 2022.

Table of Contents

1. Introduction

During the transition to post-quantum cryptography, there will be uncertainty as to the strength of cryptographic algorithms; we will no longer fully trust traditional cryptography such as RSA, Diffie-

Hellman, DSA and their elliptic curve variants, but we will also not fully trust their post-quantum replacements until they have had sufficient scrutiny. Unlike previous cryptographic algorithm migrations, the choice of when to migrate and which algorithms to migrate to, is not so clear. Even after the migration period, it may be advantageous for an entity's cryptographic identity to be composed of multiple public-key algorithms.

The deployment of composite signatures using post-quantum algorithms will face two challenges

* Algorithm strength uncertainty: During the transition period, some post-quantum signature and encryption algorithms will not be fully trusted, while also the trust in legacy public key algorithms will start to erode. A relying party may learn some time after deployment that a public key algorithm has become untrustworthy, but in the interim, they may not know which algorithm an adversary has compromised.

* Backwards compatibility: During the transition period, post-quantum algorithms will not be supported by all clients.

This document provides a mechanism to address algorithm strength uncertainty by building on ~~ reference draft-ounsworth-pq-composite-pubkeys ~~ by providing formats for encoding multiple signature values into existing public signature fields, as well as the process for validating a composite signature. Backwards compatibility is addressed via the Composite-OR mechanism described herein.

This document is intended for general applicability anywhere that digital signatures are used within PKIX and CMS structures.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are used in this document:

ALGORITHM: An information object class for identifying the type of cryptographic operation to be performed. This document is primarily concerned with algorithms for producing digital signatures.

BER: Basic Encoding Rules (BER) as defined in [X.690].

COMPONENT ALGORITHM: A single basic algorithm which is contained within a composite algorithm.

COMPOSITE ALGORITHM: An algorithm which is a sequence of two or more component algorithms, as defined in Section 2.

DER: Distinguished Encoding Rules as defined in [X.690].

LEGACY: For the purposes of this document, a legacy key or signature is a non-composite key or signature.

PUBLIC / PRIVATE KEY: The public and private portion of an asymmetric cryptographic key, making no assumptions about which algorithm.

SIGNATURE: A digital cryptographic signature, making no assumptions about which algorithm.

2. Composite Identifiers and Structures

In order for signatures to be composed of multiple algorithms, we define encodings consisting of a sequence of signature primitives (aka "component algorithms") such that these structures can be used as a drop-in replacement for existing signature fields such as those found in PKCS#10 [RFC2986], CMP [RFC4210], X.509 [RFC5280], CMS [RFC5652].

This section defines the following structures:

* The id-alg-composite is an AlgorithmIdentifier identifying a composite signature object.

  The sa-CompositeSignature AlgorithmIdentifier and the corresponding CompositeParams identify the algorithm(s) used in a composite signature.

* The CompositeSignatureValue, carries a sequence of signatures that are generated by a CompositePrivateKey, and can be verified with the corresponding CompositePublicKey.

EDNOTE 2: the choice to define composite algorithm parameters as a sequence inside the existing fields avoids the exponential proliferation of OIDs that are needed for each combination of signature algorithms in other schemes for achieving multi-key certificates. This scheme also naturally extends from 2-keypair to n-keypair keys and certificates.

EDNOTE 2a: We have heard community feedback that the ASN.1 structures presented here are too flexible in that allow arbitrary combinations of an arbitrary number of signature algorithms. The

feedback is that this is too much of a "footgun" for implementors and sysadmins. We are working on an alternative formulation using ASN.1 information object classes that allow for compiling explicit pairs of algorithmIDs. We would love community feedback on which approach is preferred. See slide 30 of this presentation: https://datatracker.ietf.org/meeting/interim-2021-lamps-01/materials/slides-interim-2021-lamps-01-sessa-position-presentation-by-mike-ounsworth-00.pdf

2.1. Algorithm Identifier

The following object identifier is used for identifying a composite signature. Additional encoding information is provided below for each of these objects.

```
id-alg-composite OBJECT IDENTIFIER ::= {
    iso(1)  identified-organization(3) dod(6) internet(1) private(4)
    enterprise(1) OpenCA(18227) Algorithms(2) id-alg-composite(1) }
```

EDNOTE 3: this is a temporary OID for the purposes of prototyping. We are requesting IANA to assign a permanent OID, see Section 5.

2.2. Composite Keys

A Composite signature MUST be associated with a Composite public key as defined in ~~ reference draft-ounsworth-pq-composite-pubkey ~~.

2.2.1. Key Usage Bits

For protocols such as X.509 [RFC5280] that specify key usage along with the public key, then the composite public key associated with a composite signature MUST have a signing-type key usage.

If the keyUsage extension is present in a Certification Authority (CA) certificate that indicates id-composite-key, then any combination of the following values MAY be present:

```
digitalSignature;
nonRepudiation;
keyCertSign; and
cRLSign.
```

If the keyUsage extension is present in an End Entity (EE) certificate that indicates id-composite-key, then any combination of the following values MAY be present:

```
digitalSignature; and
nonRepudiation;
```

2.3. Composite Signature

The ASN.1 algorithm object for a composite signature is:

```
sa-CompositeSignature SIGNATURE-ALGORITHM ::= {
    IDENTIFIER id-alg-composite
    VALUE CompositeSignatureValue
    PARAMS TYPE CompositeParams ARE required
    PUBLIC-KEYS { pk-Composite }
    SMIME-CAPS { IDENTIFIED BY id-alg-composite } }
}
```

The following algorithm parameters MUST be included:

```
CompositeParams ::= SEQUENCE SIZE (2..MAX) OF AlgorithmIdentifier
```

The signature's CompositeParams sequence MUST contain the same component algorithms listed in the same order as in the associated CompositePrivateKey and CompositePublicKey.

The output of the composite signature algorithm is the DER encoding of the following structure:

```
CompositeSignatureValue ::= SEQUENCE SIZE (2..MAX) OF BIT STRING
```

Where each BIT STRING within the SEQUENCE is a signature value produced by one of the component keys. It MUST contain one signature value produced by each component algorithm, and in the same order as in the associated CompositeParams object.

The choice of SEQUENCE OF BIT STRING, rather than for example a single BIT STRING containing the concatenated signature values, is to gracefully handle variable-length signature values by taking advantage of ASN.1's built-in length fields.

2.4. Encoding Rules

Many protocol specifications will require that composite signature data structures be represented by an octet string or bit string.

When an octet string is required, the DER encoding of the composite data structure SHALL be used directly.

When a bit string is required, the octets of the DER encoded composite data structure SHALL be used as the bits of the bit string, with the most significant bit of the first octet becoming the first bit, and so on, ending with the least significant bit of the last octet becoming the last bit of the bit string.

In the interests of simplicity and avoiding compatibility issues, implementations that parse these structures MAY accept both BER and DER.

3. Composite Signature Processes

This section specifies the processes for generating and verifying composite signatures.

This process addresses algorithm strength uncertainty by providing the verifier with parallel signatures from all the component signature algorithms; thus breaking the composite signature would require breaking all of the component signatures.

3.1. Composite Signature Generation Process

Generation of a composite signature involves applying each component algorithm's signature process to the input message according to its specification, and then placing each component signature value into the CompositeSignatureValue structure defined in Section 2.3.

The following process is used to generate composite signature values.

```
Input:
     K1, K2, .., Kn     Private keys for the n component signature
                        algorithms, a CompositePrivateKey
     M                  Message to be signed, an octet string

Output:
     S                  The signatures, a CompositeSignatureValue
```

Signature Generation Process:

1. Generate the n component signatures independently, according to their algorithm specifications.

```
for i := 1 to n
    Si := Sign( Ki, M )
```

2. Encode each component signature S1, S2, .., Sn into a BIT STRING according to its algorithm specification.

```
S ::= Sequence { S1, S2, .., Sn }
```

3. Output S

Since recursive composite public keys are disallowed in ~~ Reference draft-ounsworth-pq-composite-pubkeys sec-composite-pub-keys ~~, no component signature may itself be a composite; ie the signature

generation process MUST fail if one of the private keys K1, K2, .., Kn is a composite with the OID id-alg-composite.

A composite signature MUST produce and include in the output a signature value for every component key in the corresponding CompositePrivateKey. For this mode, please see Composite-OR in section Section 3.2.

3.2. Composite-OR Signature Generation Process

EDNOTE: This section was written with the intention of keeping the primary Composite OID reserved for the simple and strict mode; if you want to do either a simple OR, or a custom policy then we have given a different OID. We are still debating whether this is useful to specify at issuing time, or whether this is adding needless complexity to the draft.

If the algorithm ID of the public key associated with this signature is id-composite-or-key then the signer MAY use only a subset of the component keys and therefore produce fewer signatures than the number of component keys.

Composite-OR signature generation uses the same structures and algorithms as Composite, with the difference that the signature generation process may emit a null instead of a signature value in step 1 for one or more component algorithms. A Composite-OR signature MUST NOT be entirely null; it must contain at least one valid signature.

The design intent of this mode is to support migration scenarios where an end entity has been issued keys on algorithms that either itself or the peer with which it is communicating do not (yet) support. This design allows for both the mode where the signer omits signatures that it knows its peer cannot process in order to save bandwidth and performance, and the mode where it includes all component signatures and allows the verifier to choose how many to verify. The latter is RECOMMENDED for signatures that need both sort-term backwards compatibility as well as long-term security.

EDNOTE: Do we want to allow a Composite-OR with only a single signature to produce non-composite signatureAlgorithm and signatureValua as per [RFC5280]? Advantages: bandwidth savings of an extra OID and some sequences with one element. Disadvantages: ambiguous whether a signature is traditional or composite until you look at the corresponding public key.

3.3. Composite Signature Verification Process

Verification of a composite signature involves applying each component algorithm's verification process according to its specification.

In the absence of an application profile specifying otherwise, compliant applications MUST output "Valid signature" (true) if and only if all component signatures were successfully validated, and "Invalid signature" (false) otherwise.

The following process is used to perform this verification.

```
Input:
     P    Signer's composite public key
     M    Message whose signature is to be verified, an octet string
     S    Composite Signature to be verified
     A    Composite Algorithm identifier

Output:
    Validity      "Valid signature" (true) if the composite signature
                  is valid, "Invalid signature" (false) otherwise.

Signature Verification Procedure::
   1. Parse P, S, A into the component public keys, signatures,
      and algorithm identifiers

      P1, P2, .., Pn := Desequence( P )
      S1, S2, .., Sn := Desequence( S )
      A1, A2, .., An := Desequence( A )

    If Error during Desequencing, or the three sequences have
    different numbers of elements, or any of the public keys P1, P2, ..,
    algorithm identifiers A1, A2, .., An are composite with the OID
    id-alg-composite then output "Invalid signature" and stop.

   2. Check each component signature individually, according to its
       algorithm specification.
       If any fail, then the entire signature validation fails.

     for i := 1 to n
          if not verify( Pi, M, Si ), then
            output "Invalid signature"

      if all succeeded, then
        output "Valid signature"
```

Since recursive composite public keys are disallowed in ~~ Reference draft-ounsworth-pq-composite-keys sec-composite-pub-keys ~~, no component signature may be composite; ie the signature verification

procedure MUST fail if any of the public keys P1, P2, .., Pn or algorithm identifiers A1, A2, .., An are composite with the OID id-alg-composite.

3.4. Composite-OR Signature Verification

EDNOTE: This section was written with the intention of keeping the primary Composite OID reserved for the simple and strict mode; if you want to do either a simple OR, or a custom policy then we have given a different OID. We are still debating whether this is useful to specify at issuing time, or whether this is adding needless complexity to the draft.

When the public key associated with the signature being verified has algorithm id-composite-or-key, then an alternate verification processes MAY be used, at the discretion of the implementor. In this section we provide some examples of alternate verification processes.

If the signature is a traditional (non-composite) algorithm and value or a composite signature with a single component, then it MAY be considered valid if it verifies under one of the component keys.

If the signature is composite, then the implementor MAY implement policy for which combinations are acceptable.

EDNOTE: Does this mean Composite-OR end entity certificates need to be issued by a PKI that is marked as Composite-OR all the way to the top so that verifiers that do not support all the algorithms don't fail? Need to think more about the security implications of allowing a Composite-or in an end entity cert implicitely turning all Composite algIDs into Composite-or algIDs in its cert chain.

EDNOTE: Do we need to specify the semantics of verifying an "n of m" subset signature? I suspect that specifying this in general will be a rat's nest of edge cases, so I propose to "leave this to the implementor".

3.4.1. Composite-OR Legacy Mode

The Composite-OR Legacy Mode is provided to facilitate migration by allowing existing PKI entities (including root CAs, intermediate CAs, and end entities) to have their existing keys re-certified inside a Composite-OR structure along with Post-Quantum keys, and for signatures made by that key prior to the migration to remain valid. Note that Composite-OR Legacy Mode is only provided for signature verification, and not for signature generation; legacy signatures SHOULD NOT be produced from a Composite key.

EDNOTE: to further solidify this, we could add a clause that Legacy Mode signatures are to fail if the signature was produced after notBefore date of the Composite-OR certificate?

In Composite-OR Legacy Mode, a legacy signature algorithm and legacy signature value MAY be validated against a Composite-OR public key. The legacy signature algorithm is to be interpreted by the verifier as a sa-CompositeSignature with CompositeParams in the following way:

```
CompositeParams {legacyAlgorithmIdentifier, null, .., null}
```

with the correct number of nulls to match the Composite-OR public key that the signature is being verified against. For the purposes of a signature validation under Composite-OR Legacy Mode, a null AlgorithmIdentifier is considered to be a match for the corresponding algorithm in the Composite-OR public key.

The legacy signature value is to be interpreted by the verifier as a sa-CompositeSignature with CompositeParams in the following way:

```
CompositeSignatureValue  {legacySignatureValue, null, .., null}
```

with the correct number of nulls to match the Composite-OR public key that the signature is being verified against. The verification algorithm in section Section 3.4 applies.

Security consideration: when implementing Composite-OR Legacy Mode, it is important to catch the edge case of {null, null, .., null} for both AlgorithmIdentifier and SignatureValue and return Invalid Signature.

It is RECOMMENDED that Composite-OR Legacy Mode be implemented as an optional mode in the verifier that can be enabled or disabled by runtime configuration or policy.

EDNOTE: the signing public key is often identified in the signed document by issuer+serialNumber or by an SKI containing a hash of the public key value. Might need X.509 extensions identifying the SKI of the legacy cert it's replacing?

4. In Practice

This section addresses practical issues of how this draft affects other protocols and standards.

~~~ BEGIN EDNOTE 10~~~
~~~

EDNOTE 10: Possible topics to address:

- *The size of these certs and cert chains.

- *In particular, implications for (large) composite keys / signatures / certs on the handshake stages of TLS and IKEv2.

- *If a cert in the chain is a composite cert then does the whole chain need to be of composite Certs?

- *We could also explain that the root CA cert does not have to be of the same algorithms. The root cert SHOULD NOT be transferred in the authentication exchange to save transport overhead and thus it can be different than the intermediate and leaf certs.

- *We could talk about overhead (size and processing).

- *We could also discuss backwards compatibility.

- *We could include a subsection about implementation considerations.

~~~ END EDNOTE 10~~~

4.1. Cryptographic protocols

This section talks about how protocols like (D)TLS and IKEv2 are affected by this specifications. It will not attempt to solve all these problems, but it will explain the rationale, how things will work and what open problems need to be solved. Obvious issues that need to be discussed.

- *How does the protocol declare support for composite signatures? TLS has hooks for declaring support for specific signature algorithms, however it would need to be extended, because the client would need to declare support for both the composite infrastructure, as well as for the various component signature algorithms.

- *How does the protocol use the multiple keys. The obvious way would be to have the server sign using its composite public key; is this sufficient.

- *Overhead; including certificate size, signature processing time, and size of the signature.

- *How to deal with crypto protocols that use public key encryption algorithms; this document only lists how to work with signature algorithms. Encoding composite public keys is straightforward;
~~~

encoding composite ciphertexts is less so - we decided to put that off to another draft.

5. IANA Considerations

The ASN.1 module OID is TBD. The id-alg-composite OID is to be assigned by IANA. The authors suggest that IANA assign an OID on the id-pkix arc:

```
id-alg-composite OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) algorithms(6) composite(??) }
```

6. Security Considerations

6.1. Policy for Deprecated and Acceptable Algorithms

Traditionally, a public key, certificate, or signature contains a single cryptographic algorithm. If and when an algorithm becomes deprecated (for example, RSA-512, or SHA1), it is obvious that structures using that algorithm are implicitly revoked.

In the composite model this is less obvious since a single public key, certificate, or signature may contain a mixture of deprecated and non-deprecated algorithms. Moreover, implementers may decide that certain cryptographic algorithms have complementary security properties and are acceptable in combination even though neither algorithm is acceptable by itself.

Specifying a modified verification algorithm to handle these situations is beyond the scope of this draft, but could be desirable as the subject of an application profile document, or to be up to the discretion of implementers.

2. Check policy to see whether A1, A2, ..., An constitutes a valid combination of algorithms.

   ```
   if not checkPolicy(A1, A2, ..., An), then
     output "Invalid signature"
   ```

   While intentionally not specified in this document, implementors should put careful thought into implementing a meaningfull policy mechinism within the context of their signature verification engines, for example only algorithms that provide similar security levels should be combined together.

7. Appendices

7.1. ASN.1 Module

```
<CODE STARTS>

Composite-Signatures-2019
  { TBD }

DEFINITIONS IMPLICIT TAGS ::= BEGIN

EXPORTS ALL;

IMPORTS
  PUBLIC-KEY, SIGNATURE-ALGORITHM
    FROM AlgorithmInformation-2009  -- RFC 5912 [X509ASN1]
      { iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0)
        id-mod-algorithmInformation-02(58) }

  SubjectPublicKeyInfo
    FROM PKIX1Explicit-2009
      { iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0)
        id-mod-pkix1-explicit-02(51) }

  OneAsymmetricKey
    FROM AsymmetricKeyPackageModuleV1
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
        pkcs-9(9) smime(16) modules(0)
        id-mod-asymmetricKeyPkgV1(50) } ;

--
-- Object Identifiers
--

id-alg-composite OBJECT IDENTIFIER ::= { TBD }

--
-- Public Key
--

pk-Composite PUBLIC-KEY ::= {
    IDENTIFIER id-alg-composite
    KEY CompositePublicKey
    PARAMS ARE absent
    PRIVATE-KEY CompositePrivateKey
}

CompositePublicKey ::= SEQUENCE SIZE (2..MAX) OF SubjectPublicKeyInfo

CompositePrivateKey ::= SEQUENCE SIZE (2..MAX) OF OneAsymmetricKey

--
```

```
-- Signature Algorithm
--

sa-CompositeSignature SIGNATURE-ALGORITHM ::= {
    IDENTIFIER id-alg-composite
    VALUE CompositeSignatureValue
    PARAMS TYPE CompositeParams ARE required
    PUBLIC-KEYS { pk-Composite }
    SMIME-CAPS { IDENTIFIED BY id-alg-composite } }

CompositeParams ::= SEQUENCE SIZE (2..MAX) OF AlgorithmIdentifier

CompositeSignatureValue ::= SEQUENCE SIZE (2..MAX) OF BIT STRING

END

<CODE ENDS>
```

7.2. Intellectual Property Considerations

The following IPR Disclosure relates to this draft:

https://datatracker.ietf.org/ipr/3588/

8. Contributors and Acknowledgements

This document incorporates contributions and comments from a large group of experts. The Editors would especially like to acknowledge the expertise and tireless dedication of the following people, who attended many long meetings and generated millions of bytes of electronic mail and VOIP traffic over the past year in pursuit of this document:

John Gray (Entrust), Serge Mister (Entrust), Scott Fluhrer (Cisco Systems), Panos Kampanakis (Cisco Systems), Daniel Van Geest (ISARA), Tim Hollebeek (Digicert), and Francois Rousseau.

We are grateful to all, including any contributors who may have been inadvertently omitted from this list.

This document borrows text from similar documents, including those referenced below. Thanks go to the authors of those documents. "Copying always makes things easier and less error prone" - [RFC8411].

8.1. Making contributions

Additional contributions to this draft are weclome. Please see the working copy of this draft at, as well as open issues at:

https://github.com/EntrustCorporation/draft-ounsworth-composite-sigs

9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

[RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/

RFC4210, September 2005, <https://www.rfc-editor.org/info/rfc4210>.

[RFC5280] 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, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8411] Schaad, J. and R. Andrews, "IANA Registration for the Cryptographic Algorithm Object Identifier Range", RFC 8411, DOI 10.17487/RFC8411, August 2018, <https://www.rfc-editor.org/info/rfc8411>.

[X.690] ITU-T, "Information technology - ASN.1 encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ISO/IEC 8825-1:2015, November 2015.

Authors' Addresses

Mike Ounsworth
Entrust Limited
2500 Solandt Road -- Suite 100
Ottawa, Ontario K2K 3G5
Canada

Email: mike.ounsworth@entrust.com

Massimiliano Pala
CableLabs

Email: director@openca.org