Workgroup: LAMPS
Internet-Draft:
draft-ounsworth-pq-explicit-composite-keys-01
Published: 12 February 2022
Intended Status: Standards Track
Expires: 16 August 2022
Authors: M. Ounsworth S. Mister J. Gray
Entrust Entrust Entrust

Explicit Pairwise Composite Keys 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 public keys and composite signature data. This draft defines a structure generic enough to be useful beyond the post-quantum transition for any situation where a widely-supported but untrusted algorithm is being migrated to newer cryptography.

This document defines structures for binding an explicit pair of cryptographic algorithms together into a single object identifier, and it provides ASN.1 structures for encoding these pairwise composite public keys, private keys in wire protocols, as well as using them in conjunction with composite signatures, encryption and key transport mechanisms.

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 16 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 public keys and 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 providing formats for encoding multiple public keys and private keys into existing fields.

This document provides structures to encode explicit composite algorithm identifiers and parameters for use with composite signature, encryption, and key transport mechanisms defined in ~~ TODO cite corresponding drafts properly ~~.

This document is intended for general applicability anywhere that public key or private key structures are used within PKIX protocols.

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, though the public key structure could just as easily hold encryption keys.

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 one or more component algorithms, as defined in Section 2.

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

EXPLICIT COMPOSITE: Composite structures where the AlgorithmIdentifier OID explicitly defines the component algorithms. This case allows simplification and compression of the data structures.

GENERIC COMPOSITE: Composite structures that are agnostic to the choice of Algorithms that they carry.

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

PRIMITIVE PUBLIC KEY / SIGNATURE: A public key or signature object of a non-composite algorithm type.

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

2. Composite Structures

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

This section defines the following structures:

~~ TODO ~~

2.1. Composite Keys

A composite key is a single key object that performs an atomic signature or verification operation, using its encapsulated pair of component keys.

Explicit pairs can easily be defined by simply providing an OBJECT IDENTIFIER and two existing PUBLIC-KEY types to the pk-explicitComposite object class, and assigning an OID to the resulting structure. See examples of defining explicit pairs in Section 6.2.

```
-- TODO - CERT-KEY-USAGE should contain the intersection of the usages f
-- pk-explicitComposite - Composite public key information object

pk-explicitComposite{OBJECT IDENTIFIER:id, PUBLIC-KEY:firstPublicKey, Fi
    IDENTIFIER id
    KEY ExplicitCompositePublicKey{firstPublicKey, FirstPublicKeyType, s
    PARAMS ARE absent
    CERT-KEY-USAGE {digitalSignature, nonRepudiation, keyCertSign, cRLSi
}
```

The following ASN.1 object class then automatically generates the public key structure from the types defined in pk-explicitComposite.

```
-- ExplicitCompositePublicKey - The data structure for a composite publi
-- sec-alg-identifier and SecondPublicKeyType are needed because PUBLIC-
-- a set of public key types, not a single type.
-- TODO The parameters should be optional only if they are marked option


ExplicitCompositePublicKey{PUBLIC-KEY:firstPublicKey, FirstPublicKeyType
    firstPublicKey SEQUENCE {
        params firstPublicKey.&Params OPTIONAL,
        publicKey FirstPublicKeyType
    },
    secondPublicKey SEQUENCE {
        params secondPublicKey.&Params OPTIONAL,
        publicKey SecondPublicKeyType
    }
}
```

2.2. Composite Private Key

EDNOTE: THIS IS WRONG. (copied from generic draft) we need to do some work to come up with a private key structure.

The composite private key data is represented by the following structure:

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

Each element is a OneAsymmetricKey [RFC5958] object for a component private key.

The corresponding AlgorithmIdentifier for a composite private key MUST use the id-alg-composite object identifier, and the parameters field MUST be absent.

A CompositePrivateKey MUST contain at least one component private key, and they MUST be in the same order as in the corresponding CompositePublicKey.

2.3. Composite Signature

The structure pk-explicitComposite contains all the necessary information in order for the ASN.1 compiler to generate composite signature structures that are explicitely bound to the specified pair of algorithms.

EDNOTE: Is this helping, or adding complexity for no reason? In theory, explicit composite public keys can be used with generic composite signature and encryption structures (ie the SEQUENC OF model).

2.3.1. Explicit Signature Params

The following ASN.1 object class then automatically generates the signature params structure from the types defined in pk-explicitComposite.

```
-- ExplicitSignatureParams - The data structure for composite signature
-- TODO firstParams and secondParams should be optional only if they are
-- in SIGNATURE-ALGORITHM

ExplicitSignatureParams{SIGNATURE-ALGORITHM:firstAlg, SIGNATURE-ALGORITH
    firstParams firstAlg.&Params OPTIONAL,
    secondParams secondAlg.&Params OPTIONAL
}
```

EDNOTE: we need some help from the community on the ASN.1 here: "OPTIONAL" is not really the right semantics here; we really mean that they params here should be present or absent when the corresponding params are present or absent in ExplicitCompositePublicKey, which ought to be enforcable by the ASN.1 compiler, but we can't figure out the syntax for declaring that.

2.3.2. Explicit Composite Signature Algorithm

The following ASN.1 object class then automatically generates the signature algorithm structure from the types defined in pk-explicitComposite.

```
-- TODO - Would it be possible to make these definitions compatible with
-- sa-explicitCompositeSignatureAlgorithm - Composite signature algorith

sa-explicitCompositeSignatureAlgorithm{OBJECT IDENTIFIER:algId, SIGNATUR
    IDENTIFIER algId
    VALUE ExplicitCompositeSignatureValue{firstAlg.&Value, secondAlg.&Va
    PARAMS TYPE ExplicitSignatureParams{firstAlg, secondAlg} ARE require
    PUBLIC-KEYS { pk-explicitComposite{algId, firstPublicKey, FirstPubli
    SMIME-CAPS { IDENTIFIED BY algId }
}
```

2.3.3. Explicit Encryption and Key Exchange Params

~~ TODO ~~ Need analogous structures to the signature ones above.

2.4. Encoding Rules

Many protocol specifications will require that the composite public key, composite private key, and composite signature data structures be represented by an octet 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. 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~~~

3.1. PEM Storage of Composite Private Keys

CompositePrivateKeys can be encoded to the PEM format by placing a CompositePrivateKey into the privateKey field of a PrivateKeyInfo or OneAsymmetricKey object, and then applying the PEM encoding rules as defined in [RFC7468] section 10 and 11 for plaintext and encrypted private keys, respectively.

3.2. Asymmetric Key Packages (CMS)

The Cryptographic Message Syntax (CMS), as defined in [RFC5652], can be used to digitally sign, digest, authenticate, or encrypt the asymmetric key format content type.

When encoding composite private keys, the privateKeyAlgorithm in the OneAsymmetricKey SHALL be set to id-alg-composite.

The parameters of the privateKeyAlgorithm SHALL be a sequence of AlgorithmIdentifier objects, each of which are encoded according to the rules defined for each of the different keys in the composite private key.

The value of the privateKey field in the OneAsymmetricKey SHALL be set to the DER encoding of the SEQUENCE of private key values that make up the composite key. The number and order of elements in the sequence SHALL be the same as identified in the sequence of parameters in the privateKeyAlgorithm.

The value of the publicKey (if present) SHALL be set to the DER encoding of the corresponding CompositePublicKey. If this field is
~~~

present, the number and order of component keys MUST be the same as identified in the sequence of parameters in the privateKeyAlgorithm.

The value of the attributes is encoded as usual.

3.3. 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.

4. IANA Considerations

This draft does not define any OIDs, however derivative drafts that define concrete algorithm pairs will. The authors suggest that IANA assign OIDs for explicit composite pairs on the id-pkix arc under a composite() arc.

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

5. Security Considerations

5.1. 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.

5.2. Protection of Private Keys

Structures described in this document do not protect private keys in any way unless combined with a security protocol or encryption properties of the objects (if any) where the CompositePrivateKey is used (see next Section).

Protection of the private keys is vital to public key cryptography. The consequences of disclosure depend on the purpose of the private key. If a private key is used for signature, then the disclosure allows unauthorized signing. If a private key is used for key management, then disclosure allows unauthorized parties to access the managed keying material. The encryption algorithm used in the encryption process must be at least as 'strong' as the key it is protecting.

5.3. Checking for Compromised Key Reuse

CA implementations need to be careful when checking for compromised key reuse, for example as required by WebTrust regulations; when checking for compromised keys, you MUST unpack the CompositePublicKey structure and compare individual component keys. In other words, when marking a key as revoked for key compromise,

the individual component keys should be marked, not the composite key as a whole.

6. Appendices

6.1. 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
    CERT-KEY-USAGE
        { digitalSignature, nonRepudiation, keyCertSign, cRLSign }
    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>
```

6.2. Examples of defining explicit pairs

To add support for a new pair of algorithms, all that is required is the following two constructs:

```
id-sa-entrust-sha256RSAandECDSA OBJECT IDENTIFIER ::= { 1 2 3 4 }

sa-entrust-sha256RSAandECDSA SIGNATURE-ALGORITHM ::= sa-explicitComposit
    id-sa-entrust-sha256RSAandECDSA,
    sa-sha256WithRSAEncryption,
    pk-rsa,
    RSAPublicKey,
    sa-ecdsaWithSHA256,
    pk-ec,
    ECPoint
}
```

TODO: run this through an ASN.1 compiler and list here what the final generated structures look like.

6.3. Intellectual Property Considerations

The following IPR Disclosure relates to this draft:

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

7. 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 Datacard), Serge Mister (Entrust Datacard), Scott Fluhrer (Cisco Systems), Panos Kampanakis (Cisco Systems), Daniel Van Geest (ISARA), and Tim Hollebeek (Digicert).

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. References

8.1. Normative References

[RFC1421] Linn, J., "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures", RFC 1421, DOI 10.17487/RFC1421, February 1993, <https://www.rfc-editor.org/info/rfc1421>.

[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>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[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>.

[RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, DOI 10.17487/RFC5958, August 2010, <https://www.rfc-editor.org/info/rfc5958>.

[RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", RFC 7468, DOI 10.17487/

RFC7468, April 2015, <https://www.rfc-editor.org/info/rfc7468>.

[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.

8.2. Informative References

[I-D.ounsworth-pq-composite-sigs] Ounsworth, M. and M. Pala, "Composite Keys and Signatures For Use In Internet PKI", Work in Progress, Internet-Draft, draft-ounsworth-pq-composite-sigs-03, 28 July 2020, <http://www.ietf.org/internet-drafts/draft-ounsworth-pq-composite-sigs-03.txt>.

Authors' Addresses

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

Email: mike.ounsworth@entrust.com

Serge Mister
Entrust Limited
1000 Innovation Drive
Ottawa, Ontario K2K 1E3
Canada

Email: serge.mister@entrust.com

John Gray
Entrust Limited
1000 Innovation Drive
Ottawa, Ontario
Canada

Email: john.gray@entrust.com