Workgroup: LAMPS
Internet-Draft:
draft-ounsworth-pq-composite-sigs-07
Published: 8 June 2022
Intended Status: Standards Track
Expires: 10 December 2022
Authors: M. Ounsworth   M. Pala
         Entrust   CableLabs

Composite Signatures For Use In Internet PKI

Abstract

The migration to post-quantum cryptography is unique in the history of modern digital cryptography in that neither the old outgoing nor the new incoming algorithms are fully trusted to protect data for the required data lifetimes. The outgoing algorithms, such as RSA and elliptic curve, may fall to quantum cryptanalysis, while the incoming post-quantum algorithms face uncertainty about both the underlying mathematics as well as hardware and software implementations that have not had sufficient maturing time to rule out classical cryptanalytic attacks and implementation bugs.

Cautious implementer may wish to layer cryptographic algorithms such that an attacker would need to break all of them in order to compromise the data being protected. For digital signatures, this is referred to as "dual", and for encryption key establishment this as referred to as "hybrid". This document, and its companions, defines a specific instantiation of the dual and hybrid paradigm called "composite" where multiple cryptographic algorithms are combined to form a single key, signature, or key encapsulation mechanism (KEM) such that they can be treated as a single atomic object at the protocol level.

EDNOTE: the terms "dual" and "hybrid" are currently in flux. We anticipate an Informational draft to normalize terminology, and will update this draft accordingly.

This document defines the structures CompositeSignatureValue, and CompositeParams, which are sequences of the respective structure for each component algorithm. The generic composite variant is defined which allows arbitrary combinations of signature algorithms to be used in the CompositeSignatureValue and CompositeParams structures without needing the combination to be pre-registered or pre-agreed. The explicit variant is also defined which allows for a set of signature algorithm identifier OIDs to be registered together as an explicit composite signature algorithm and assigned an OID.

This document is intended to be coupled with corresponding documents that define the structure and semantics of composite public and private keys and encryption [I-D.draft-ounsworth-pq-composite-keys-01], however may also be used with non-composite keys, such as when a protocol combines multiple certificates into a single cryptographic operation.

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 10 December 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Changes in version -07
2. Introduction
  2.1. Terminology
3. Composite Signature Structures
  3.1. Composite Keys
    3.1.1. Key Usage Bits
  3.2. sa-CompositeSignature
  3.3. CompositeSignatureValue
  3.4. Encoding Rules

4. Algorithm Identifiers
  4.1. id-alg-composite (Generic Composite Signatures)
  4.2. Explicit Composite Signatures
5. Composite Signature Processes
  5.1. Composite Signature Generation Process
  5.2. Composite Signature Verification Process
6. Implementation Considerations
  6.1. Backwards Compatibility
    6.1.1. OR modes
    6.1.2. Parallel PKIs
7. IANA Considerations
8. Security Considerations
  8.1. Policy for Deprecated and Acceptable Algorithms
  8.2. OR Modes
    8.2.1. Subset Signature Generation
    8.2.2. Subset Signature Verification
9. References
  9.1. Normative References
  9.2. Informative References
Appendix A. Work in Progress
  A.1. Combiner modes (KofN)
Appendix B. Creating explicit combinations
Appendix C. Examples
  C.1. Generic Composite Signature Examples
  C.2. Explicit Composite Signature Examples
Appendix D. ASN.1 Module
Appendix E. Intellectual Property Considerations
Appendix F. Contributors and Acknowledgements
  F.1. Making contributions
Authors' Addresses

1. Changes in version -07

*Merged Generic Composite (Section 4.1) and Explicit Composite (Section 4.2) into one document and made them share a wire encoding (only differing by the OIDs used).

*Removed Composite-OR signature mode.

*Added Section 6.1 addressing backwards compatibility and ease of migration concerns.

*Added CompositeParams := Alg1, Alg2, .. Algn as an input parameter to the sig gen and verification processes.

TODO diff this against the public version and see if there are any more changes.

2. 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 and time to discover and fix implementation bugs. 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 concerns by building on [draft-ounsworth-pq-composite-keys-00] (NOTE: need kramdown formatting help with this ref) 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 using composite in conjunction with a non-composite hybrid mode such as that described in [draft-becker-guthrie-noncomposite-hybrid-auth-00] (NOTE: need kramdown formatting help with this ref).

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

2.1. 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: A standardized cryptographic primitive, as well as any ASN.1 structures needed for encoding data and metadata needed to use the algorithm. This document is primarily concerned with algorithms for producing digital signatures.

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

CLIENT: Any software that is making use of a cryptographic key. This includes a signer, verifier, encrypter, decrypter.

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 3.

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

LEGACY: For the purposes of this document, a legacy algorithm is any cryptographic algorithm currently is use which is not believe to be resistant to quantum cryptanalysis.

PKI: Public Key Infrastructure, as defined in [RFC5280].

POST-QUANTUM ALGORITHM: Any cryptographic algorithm which is believed to be resistant to classical and quantum cryptanalysis, such as the algorithms being considered for standardization by NIST.

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.

STRIPPING ATTACK: An attack in which the attacker is able to downgrade the cryptographic object to an attacker-chosen subset of original set of component algorithms in such a way that it is not detectable by the receiver. For example, substituting a composite public key or signature for a version with fewer components.

3. Composite Signature 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].

3.1. Composite Keys

A composite signature MAY be associated with a composite public key as defined in [draft-ounsworth-pq-composite-keys-00] (NOTE: need kramdown formatting help with this ref), but MAY also be associated with multiple public keys from different sources, for example multiple X.509 certificates, or multiple cryptographic modules. In the latter case, composite signatures MAY be used as the mechanism for carrying multiple signatures in a non-composite authentication mechanism such as those described in [draft-becker-guthrie-noncomposite-hybrid-auth-00] (NOTE: need kramdown formatting help with this ref).

3.1.1. 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;
```

3.2. sa-CompositeSignature

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

```
sa-CompositeSignature SIGNATURE-ALGORITHM ::= {
    IDENTIFIER identifier
    VALUE CompositeSignatureValue
    PARAMS ANY DEFINED BY ALGORITHM
    PUBLIC-KEYS { pk-Composite }
    SMIME-CAPS { IDENTIFIED BY id-alg-composite } }
}
```

The identifier specifies the type of composite signature and the component algorithms. This document defines a generic composite algorithm, identified by id-alg-composite, in Section 4.1, and allows for other standards that will define explicit algorithms that specify which component algorithms are to be contained within them.

3.3. CompositeSignatureValue

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.

A CompositeSignatureValue MUST contain the same number of component signatures as the corresponding public and private keys, and the order of component signature values MUST correspond to the component public keys.

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.

3.4. 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.

EDNOTE: will this definition include an ASN.1 tag and length byte inside the OCTET STRING object? If so, that's probably an extra unnecessary layer.

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.

4. Algorithm Identifiers

This section defines the algorithm identifier for generic composite, as well as a framework for defining explicit combinations. This section is not intended to be exhaustive and other authors may define others so long as they are compatible with the structures and processes defined in this and companion public and private key documents.

Some use-cases desire the flexibility for clients to use any combination of supported algorithms, while others desire the rigidity of explicitly-specified combinations of algorithms.

4.1. id-alg-composite (Generic Composite Signatures)

The id-alg-composite object identifier is used for identifying a generic composite signature. This algorithm allows arbitrary combinations of signature algorithms to be used in the CompositeSignatureValue and CompositeParams structures without needing the combination to be pre-registered or pre-agreed. This identifier MUST be used in sa-CompositeSignature.identifier.

```
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: this is a temporary OID for the purposes of prototyping. We are requesting IANA to assign a permanent OID, see Section 7.

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 CompositePublicKey.

The motivation for this variant is primarily for prototyping work prior to the standardization of algorithm identifiers for explicit combinations of algorithms. However, the authors envision that this variant will remain relevant beyond full standardization for example in environments requiring very high levels of crypto agility, for example where clients support a large number of algorithms or where a large number of keys will be used at a time and it is therefore prohibitive to define algorithm identifiers for every combination of pairs, triples, quadruples, etc of algorithms.

4.2. Explicit Composite Signatures

This variant provides a rigid way of specifying supported combinations of algorithms.

The motivation for this variant is to make it easier to reference and enforce specific combinations of algorithms. The authors envision this being useful for client-server negotiated protocols, protocol designers who wish to place constraints on allowable algorithm combinations in the protocol specification, as well as audited environments that wish to prove that only certain combinations will be supported by clients.

Explicit algorithms must define a new signature algorithm which consists of:

* A new algorithm identifier OID for the explicit algorithm.
* The algorithm identifier OID and PUBLIC-KEY type of each component algorithm.
* Signature parameters either declared ABSENT, or defined with a type and encoding.

See Appendix B for guidance on creating and registering OIDs for specific explicit combinations.

For explicit algorithms, it is not necessary to carry a CompositeParams with the list of component algorithms in the signature algorithm parameters because clients can infer the expected component algorithms from the algorithm identifier. The PARAMS is left optional because some types of component algorithms will require parameters to be carried, such as RSASSA-PSS-params as defined in [RFC8017]. Section 3.2 defines PARAMS ANY DEFINED BY ALGORITHM so that explicit algorithms may define params as ABSENT, use CompositeParams defined in Section 4.1 or use any other encoding that is appropriate.

In this variant, the signature is encoded as defined in Section 3.2, however the sa-CompositeSignature.identifier SHALL be an OID which is registered to represent a specific combination of component signature algorithms. See Appendix C for examples.

5. 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 forging the composite signature would require forging all of the component signatures.

5.1. 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 3.2.

The following process is used to generate composite signature values.

```
Input:
     K1, K2, .., Kn     Signing private keys. See note below on
                        composite inputs.

     A1, A2, ... An     Component signature algorithms. See note below o
                        composite inputs.

     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, Ai, 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
```

Note on composite inputs: the method of providing the list of component keys and algorithms is flexible and beyond the scope of this pseudo-code, for example they may be carried in CompositePrivateKey and CompositeParams structures. It is also possible to generate a composite signature that combines signatures from distinct keys stored in separate software or hardware keystores. Variations in the process to accommodate particular private key storage mechanisms are considered to be conformant to this document so long as it produces the same output as the process sketched above.

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 and include in the output, a signature value for every component key in the corresponding CompositePublicKey, and they MUST be in the same order; ie in the output, S1 MUST correspond to K1, S2 to K2, etc. The authors recognize that there may be valid use cases for "subset signature generation"; see Section 8.2.1 for further discussion of security implications, and Section 6.1 for further discussion of backwards compatibility implications.

For security when using a generic composite signature algorithm as defined in Section 4.1, the list of component signature algorithms A1, A2, .., An, which may be carried in a CompositeParams object, SHOULD be included in the signed message M to prevent an attacker from substituting a weaker algorithm which is compatible with the same public key. This attack is not unique or new to the composite format.

5.2. 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:
     P1, P2, .., Pn     Public verification keys. See note below on
                        composite inputs.

     M                  Message whose signature is to be verified,
                        an octet string

     S1, S2, .., Sn    Component signature values to be verified.
                       See note below on composite inputs.

     A1, A2, ... An     Component signature algorithms. See note
                        below on composite inputs.

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

Signature Verification Procedure::
   1. Check keys, signatures, and algorithms lists for consistency.

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

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

Note on composite inputs: the method of providing the list of component keys, algorithms and signatures is flexible and beyond the scope of this pseudo-code, for example they may be carried in CompositePublicKey, CompositeParams, and compositesignaturevalue structures. It is also possible to verify a composite signature where the component public verification keys belong, for example, to separate X.509 certificates or cryptographic modules. Variations in the process to accommodate particular public verification key storage mechanisms are considered to be conformant to this document so long as it produces the same output as the process sketched above.

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.

Some verification clients may include a policy mechanism for specifying acceptable subsets of algorithms. In these cases, implementer MAY, in the interest of performance of compatibility, modify the above process to skip one or more signature validations as per their local client policy. See Section 8.2 for a discussion of associated risks.

In the absence of such a policy mechanism that can be easily updated to reflect new cryptanalytic breakthroughs, clients MUST perform signature verifications in the AND mode defined here. See Section 8.2.1 for further discussion of security implications of subset signature verifications, and Section 6.1 for further discussion of backwards compatibility implications.

6. Implementation Considerations

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~~~

6.1. Backwards Compatibility

As noted in the introduction, the post-quantum cryptographic migration will face challenges in both ensuring cryptographic strength against adversaries of unknown capabilities, as well as providing ease of migration. The composite mechanisms defined in this document primarily address cryptographic strength, however this section contains notes on how backwards compatibility may be obtained.

The term "ease of migration" is used here to mean that existing systems can be gracefully transitioned to the new technology without requiring large service disruptions or expensive upgrades. The term "backwards compatibility" is used here to mean something more specific; that existing systems as they are deployed today can interoperate with the upgraded systems of the future.

These migration and interoperability concerns need to be thought about in the context of various types of protocols that make use of X.509 and PKIX with relation to digital signature objects, from online negotiated protocols such as TLS 1.3 [RFC8446] and IKEv2 [RFC7296], to non-negotiated asynchronous protocols such as S/MIME signed email [RFC8551], document signing such as in the context of the European eIDAS regulations [eIDAS2014], and publicly trusted code signing [codeSigningBRsv2.8], as well as myriad other standardized and proprietary protocols and applications that leverage CMS [RFC5652] signed structures.

6.1.1. OR modes

Section 5.1 and Section 5.2 make reference to subset signature generation and verification modes to achieve an OR relation between component signatures, where senders and / or receivers are permitted to ignore some component keys. Some envisioned uses of this include environments where the client encounters a component signature algorithm for which it does not posses a compatible implementation but wishes to proceed with the signature verification using the subset of component signatures for which it does have compatible implementations. Such a mechanism could be designed to provide ease of migration by allowing for composite keys to be distributed and used before all clients in the environment are fully upgraded, but it does not allow for full backwards compatibility since clients would at least need to be upgraded from their current state to be able to parse the composite structures.
~~~

6.1.2. Parallel PKIs

We present the term "Parallel PKI" to refer to the setup where a PKI end entity possesses two or more distinct public keys or certificates for the same identity (name), but containing keys for different cryptographic algorithms. One could imagine a set of parallel PKIs where an existing PKI using legacy algorithms (RSA, ECC) is left operational during the post-quantum migration but is shadowed by one or more parallel PKIs using pure post quantum algorithms or composite algorithms (legacy and post-quantum).

Equipped with a set of parallel public keys in this way, a client would have the flexibility to choose which public key(s) or certificate(s) to use in a given signature operation.

For negotiated protocols, the client could choose which public key(s) or certificate(s) to use based on the negotiated algorithms, or could combine two of the public keys for example in a non-composite hybrid method such as [draft-becker-guthrie-noncomposite-hybrid-auth-00] (NOTE: need kramdown formatting help with this ref) or [draft-guthrie-ipsecme-ikev2-hybrid-auth-00]. Note that it is possible to use the signature algorithms defined in Section 4 as a way to carry the multiple signature values generated by one of the non-composite public mechanism in protocols where it is easier to support the composite signature algorithms than to implement such a mechanism in the protocol itself. There is also nothing precluding a composite public key from being one of the components used within a non-composite authentication operation; this may lead to greater convenience in setting up parallel PKI hierarchies that need to service a range of clients implementing different styles of post-quantum migration strategies.

For non-negotiated protocols, the details for obtaining backwards compatibility will vary by protocol, but for example in CMS [RFC5652], the inclusion of multiple SignerInfo objects is often already treated as an OR relationship, so including one for each of the signer's parallel PKI public keys would, in many cases, have the desired effect of allowing the receiver to choose one they are compatible with and ignore the others, thus achieving full backwards compatibility.

7. 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(??) }
```

8. Security Considerations

8.1. 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 clients performing signature verifications should be updated to fail to validate signatures using these algorithms.

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"
```

8.2. OR Modes

8.2.1. Subset Signature Generation

This document defines a composite signature generation process in Section 5.1 where the signer MUST produce a signature value with each of their component private keys, this providing full protection of the content under all available component algorithms.

The authors recognize that there may be cases where a client may wish to generate a composite signature that only uses a subset of the available component algorithms, for example to save bandwidth, or because a client has been issued a key for which it does not (yet) have implementations of all component algorithms. This could be easily encoded by placing a NULL value into the corresponding field of the CompositeSignatureValue. However, this mode was intentionally omitted from this specification as it trivially allows for stripping attacks where an attacker replaces a valid component signature value with NULL, thus reducing the security of the composite signature to the weakest of the available component algorithms.

Implementer who wish to perform subset signature generations are advised to couple it with an out-of-band policy mechanism that limits the potential for stripping attacks. Note that, in an effort to keep compliant implementations simple and secure, implementations claiming to be compliant with this draft MUST NOT generate subset signatures in this way, and MUST reject during verification any subset signatures that they encounter.

8.2.2. Subset Signature Verification

This document defines a composite signature verification process in Section 5.2 where the verifier verifies all component signatures and fails if any component fails. The authors recognize that there will be scenarios where the verifier considers a single component algorithm -- or subset of component algorithms -- to provide sufficient security, and therefore for performance reasons wishes to skip the verification of one or more component signatures.

-- harmonize this with Serge's blurb --

Implementers who wish to perform subset signature verifications are advised to couple it with an out-of-band policy mechanism that can control the list of acceptable algorithm combinations, and keep this list up to date as new cryptanalytic advances are made.

Risks:

*Failing to update client verification policy in response to advances in cryptanalysis

*Verifications of a subset of signatures leads to ambiguity in the security strength of the signature verification; ie if a message carries two signatures, one at 128 bits and the other at 112 bits of security and clients are verifying in an OR mode with flexible policy, then it becomes difficult to audit the security strength used at runtime.

*Moreover, verifying multiple algorithms provides security even in the event that one of the algorithms has already been broken, but knowledge of the break has not been made public yet.

9. References

9.1. 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.

9.2. Informative References

[Bindel2017] Bindel, N., Herath, U., McKague, M., and D. Stebila, "Transitioning to a quantum-resistant public key infrastructure", 2017, <https://link.springer.com/chapter/10.1007/978-3-319-59879-6_22>.

[I-D.becker-guthrie-noncomposite-hybrid-auth] Becker, A., Guthrie, R., and M. J. Jenkins, "Non-Composite Hybrid Authentication in PKIX and Applications to Internet

Protocols", Work in Progress, Internet-Draft, draft-becker-guthrie-noncomposite-hybrid-auth-00, 22 March 2022, <https://www.ietf.org/archive/id/draft-becker-guthrie-noncomposite-hybrid-auth-00.txt>.

[I-D.ounsworth-pq-composite-keys] Ounsworth, M. and M. Pala, "Composite Public and Private Keys For Use In Internet PKI", Work in Progress, Internet-Draft, draft-ounsworth-pq-composite-keys-00, 12 July 2021, <https://www.ietf.org/archive/id/draft-ounsworth-pq-composite-keys-00.txt>.

[RFC3279] Bassham, L., Polk, W., and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, DOI 10.17487/RFC3279, April 2002, <https://www.rfc-editor.org/info/rfc3279>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

Appendix A. Work in Progress

A.1. Combiner modes (KofN)

For content commitment use-cases, such as legally-binding non-repudiation, the signer (whether it be a CA or an end entity) needs to be able to specify how its signature is to be interpreted and verified.

For now we have removed combiner modes (AND, OR, KofN) from this draft, but we are still discussing how to incorporate this for the cases where it is needed (maybe a X.509 v3 extension, or a signature algorithm param).

Appendix B. Creating explicit combinations

The following ASN.1 Information Objects may be useful in defining and parsing explicit pairs of signature algorithms.

... TODO ... copy & adapt from the keys draft.

Appendix C. Examples

C.1. Generic Composite Signature Examples

TODO

C.2. Explicit Composite Signature Examples

TODO

Appendix D. 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>
```

Appendix E. Intellectual Property Considerations

The following IPR Disclosure relates to this draft:

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

Appendix F. 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].

F.1. Making contributions

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

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

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