Updated Parameterized SIGNED ASN.1 Type for X.509 (PKIX)

draft-vangeest-lamps-update-asn1-signed-type-00

Daniel Van Geest, Carl Wallace

IETF 126 – LAMPS

Problem

- CONTAINING clause in SIGNED{ToBeSigned} can refer to an absent type

RFC 5912

```
SIGNED{ToBeSigned} ::= SEQUENCE {
     toBeSigned           ToBeSigned,
     algorithmIdentifier  SEQUENCE {
         algorithm        SIGNATURE-ALGORITHM.
                            &id({SignatureAlgorithms}),
         parameters       SIGNATURE-ALGORITHM.
                            &Params({SignatureAlgorithms}
                              {@algorithmIdentifier.algorithm}) OPTIONAL
     },
     signature BIT STRING (CONTAINING SIGNATURE-ALGORITHM.&Value(
                              {SignatureAlgorithms}
                              {@algorithmIdentifier.algorithm}))
  }
```

RFC 5912

```
SIGNATURE-ALGORITHM ::= CLASS {
    &id               OBJECT IDENTIFIER UNIQUE,
    &Value            OPTIONAL,
    &Params           OPTIONAL,
    &paramPresence    ParamOptions DEFAULT absent,
    &HashSet          DIGEST-ALGORITHM OPTIONAL,
    &PublicKeySet     PUBLIC-KEY OPTIONAL,
    &smimeCaps        SMIME-CAPS OPTIONAL
} WITH SYNTAX {
…
}
```

draft-ietf-lamps-pq-composite-sigs

```
sa-CompositeSignature{OBJECT IDENTIFIER:id,
     PUBLIC-KEY:publicKeyType }
        SIGNATURE-ALGORITHM ::=  {
           IDENTIFIER id
           -- VALUE no ASN.1 wrapping --
           PARAMS ARE absent
           PUBLIC-KEYS {publicKeyType}
           SMIME-CAPS { IDENTIFIED BY id }
        }
```

And ML-DSA, SLH-DSA, HSS-LMS, XMSS, Ed25519, RSA, sa-unsigned

RFC 5912

```
SIGNED{ToBeSigned} ::= SEQUENCE {
     toBeSigned           ToBeSigned,
     algorithmIdentifier  SEQUENCE {
         algorithm        SIGNATURE-ALGORITHM.
                            &id({SignatureAlgorithms}),
         parameters       SIGNATURE-ALGORITHM.
                            &Params({SignatureAlgorithms}
                              {@algorithmIdentifier.algorithm}) OPTIONAL
     },
     signature BIT STRING (CONTAINING UNSPECIFIED(
                              {SignatureAlgorithms}
                              {@algorithmIdentifier.algorithm}))
  }
```

Problem

- It turns out this is not valid ASN.1
- ASN.1 syntax checkers and compilers let this slide because the type might be present
- Decoders compiled from these modules explode when parsing content for objects where the type is absent

Fix

- PKIX1Explicit-2026 module: PKIX1Explicit-2009 without constraints on SIGNED.signature

```
SIGNED{ToBeSigned} ::= SEQUENCE {
      toBeSigned          ToBeSigned,
      algorithmIdentifier AlgorithmIdentifier{SIGNATURE-ALGORITHM,
                          {SignatureAlgorithms}},
      signature           BIT STRING
  }
```

- No more constraints for DSA, ECDSA, sa-noSignature
  - Code will have to deal with this now

Fix (Alternate)

```
SIGNED{ToBeSigned} ::= SEQUENCE {
       toBeSigned          ToBeSigned,
       algorithmIdentifier AlgorithmIdentifier{SIGNATURE-ALGORITHM,
                           {SignatureAlgorithms}},
       signature           BIT STRING (CONSTRAINED BY { ToBeSigned  -- with the following processing:
                               -- if SIGNATURE-ALGORITHM.&Value {
                               -- @algorithmIdentifier.algorithmIdentifier}
                               -- is specified, then the content of the BIT STRING
                               -- is the DER-encoded &Value type representing the
                               -- signature over ToBeSigned
                               -- (i.e., equivalent to a (CONTAINING &Value ENCODED BY der) clause)
                               -- Otherwise if &Value is unspecified, the content is the array of octets
                               -- representing the opaque (i.e., not ASN1) encoded signature with no
                               --  leading type or length
                               })}
```

Fix

- RFC 5958's ASN.1 module had a commented-out option for OneAsymmetricKey with the same problem
  - Remove this entirely

Next

- This problem has been encountered in the real world (someone else's Composite ML-DSA implementation)
- WG Adoption?