The devil is in the details

Migration strategies for cryptographic algorithms
(Migration to quantum-safe algorithms)

Erik Andersen
Andersen's L-Service
Member of the Danish TC57 committee & WG15
Member of ITU-T Study Group 17, Question 11
era@x500.eu
Interworking issue

- When moving to new cryptographic algorithms not everybody will move at the same time (no big bang)
- Will potentially create interworking problems
- **Solution:** During a migration period duplicate the cryptographic algorithm(s)
The strategy

At the start of the migration period, two cryptographic algorithms of a particular type are allowed to be included in specific way

- The **native** algorithm used before start of migration period together with associated information
- An **alternative**, assumingly stronger algorithm together with associated information

- An advanced entity will include both algorithms
- A back level entity will include only the native algorithm
The alternative algorithm and associated information shall be specified in such a way that a back level recipient can ignore it.

A back level recipient will ignore the alternative algorithm, but validate according to the native one.

An advanced recipient will validate according to the alternative algorithm.

At the end of the migration period, the alternative algorithm becomes the new native algorithm.
Two different cases

X.509 data types with extension mechanism:
- Public-key certificates
- Attribute certificates
- Certificate Revocation Lists (CRLs)
- Authorization and Validation Lists (AVLs)

Communications protocols
Extension characteristics

A supported extension shall be processed

An unsupported non-critical extension shall be ignored

These characteristics are utilized for migration purposes
## Public-key certificate

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td></td>
</tr>
<tr>
<td>Serial Number</td>
<td></td>
</tr>
<tr>
<td>Issuer signature algorithm</td>
<td></td>
</tr>
<tr>
<td>Issuer</td>
<td></td>
</tr>
<tr>
<td>Validity</td>
<td></td>
</tr>
<tr>
<td>Subject</td>
<td></td>
</tr>
<tr>
<td>Subject’s public key info</td>
<td></td>
</tr>
<tr>
<td>Issuer Unique Id</td>
<td></td>
</tr>
<tr>
<td>Subject Unique Id</td>
<td></td>
</tr>
<tr>
<td>Extensions</td>
<td></td>
</tr>
<tr>
<td>Signature algorithm</td>
<td></td>
</tr>
<tr>
<td>Signature</td>
<td></td>
</tr>
<tr>
<td>Extension 1</td>
<td></td>
</tr>
<tr>
<td>Extension 2</td>
<td></td>
</tr>
<tr>
<td>Alternative Signature algorithm</td>
<td></td>
</tr>
<tr>
<td>Alternative subject’s public key info</td>
<td></td>
</tr>
<tr>
<td>Alternative Signature</td>
<td></td>
</tr>
<tr>
<td>Extension n</td>
<td></td>
</tr>
</tbody>
</table>
Protocol migration considerations

- When cryptographic algorithms may be negotiated before its use
- When cryptographic algorithms cannot be negotiated before its use
- Special considerations for new protocols
- Special considerations existing protocols
- Special considerations whether ASN.1 extension marks are supported or not
- Special considerations for digital signatures
Use or non-use of handshake

A formal handshake exchange before the data transfer phase allows:

- Cryptographic algorithms not used during the handshake can be negotiated.
- Cryptographic algorithms used during the handshake require special measures for migration.

Non-use of handshake requires special measures for all used cryptographic algorithms.
A sending entity provides a sequence of cryptographic algorithm alternatives in a handshake request.

The recipient shall take the first top one it supports to be included in the handshake accept.

During the migration period:

- The sender puts the alternative cryptographic algorithm at the top and the native cryptographic algorithm as number two.
- If the recipient supports the alternative algorithm, that is returned in the accept. Otherwise, it takes the second alternative.

Should be clearly specified in the English text.
Where negotiation not possible

- Protocols without handshake
- Cryptographic algorithms used during the handshake
Adding alternative algorithm to existing association request when extension marks are used

Before:

```plaintext
HandshakeReq ::= SEQUENCE {
    ---crypt    AlgorithmIdentifier{{SupportedSymmetricKeyAlgorithms}},
    ---...
    ...
}
```

After:

```plaintext
HandshakeReq ::= SEQUENCE {
    ---crypt    AlgorithmIdentifier{{SupportedSymmetricKeyAlgorithms}},
    ---...
    ..., 
    altCrypt AlgorithmIdentifier{{SupportedSymmetricKeyAlgorithms}}
}
```
Adding alternative algorithm to existing association request when extension marks are not used

The parameter field of an algorithm specifications may be any data type. This allows for defining multiple algorithms within a single algorithm specification

```
multipleSignaturesAlgo ALGORITHM ::= {
  PARMS            MultipleSignaturesAlgo
  IDENTIFIED BY   id-algo-multipleEncryptionAlgo }

MultipleSignaturesAlgo ::= SEQUENCE SIZE (1..MAX) OF
  algo            SEQUENCE {
    algorithm ALGORITHM.&id({SupportedSignatureAlgorithms}),
    parameter ALGORITHM.&Type({SupportedSignatureAlgorithms}{@algorithm}) OPTIONAL,
    ...  }
```

**Pro:** No change to the specification

**Con:** Change to implementations

The necessary specification, including object identifiers, are defined in Rec. ITU-T X.510 | IEC 9594-11
SIGNED{ToBeSigned} ::= SEQUENCE {
  ToBeSigned, ToBeSigned,
  AlgorithmIdentifier{{SupportedAlgorithms}},
  BIT STRING,
  ...
}
Alt. Signature on APDU for new protocols

Extended SIGNED data type

SIGNED{ToBeSigned} ::= SEQUENCE {
  toBeSigned        ToBeSigned,
  algorithmIdentifier AlgorithmIdentifier{{SupportedAlgorithms}},
  signature         BIT STRING,
  ...,
  altAlgorithmIdentifier AlgorithmIdentifier{{SupportedAlgorithms}} OPTIONAL,
  altSignature      BIT STRING OPTIONAL }

APDU

Protected by signature

Signature algorithm
Alt. signature algorithm

Ignore if algorithm not supported

ToBeSigned

Extension mark

Optional - to be ignored if algorithm not supported

Signed{ToBeSigned} ::= SEQUENCE {
  toBeSigned ToBeSigned,
  signature BIT STRING,
  ...,
  altSignature BIT STRING OPTIONAL }
Alt. Signature on Application Protocol Data Unit (APDU) - 2

Request:

```plaintext
---
sigAlg       AlgorithmIdentifier {{SupportedSignatureAlgorithms}},
altSigAlg    [0] AlgorithmIdentifier {{SupportedSignatureAlgorithms}} OPTIONAL,
---
```

Response:

```plaintext
---
sigSel       CHOICE {
  sigAlg       AlgorithmIdentifier {{SupportedSignatureAlgorithms}},
  altSigAlg    [0] AlgorithmIdentifier {{SupportedSignatureAlgorithms}}
},
---
```

The responder is able to indicate that it has migrated, which might be used in subsequent communications.
Alt. Signature on APDU for an existing protocols with extension marks supported

APDU

Protected by signature

Signature algorithm

ToBeSigned

Alt. signature algorithm

Signature algorithm

Signature

Alt. signature algorithm

Alt. Signature

Protected by signature

Extension marks

Optional - to be ignored

SIGNED{ToBeSigned} ::= SEQUENCE {
  toBeSigned           ToBeSigned,
  algorithmIdentifier  AlgorithmIdentifier{{SupportedAlgorithms}},
  signature            BIT STRING,
  ...,                 
  altAlgorithmIdentifier  AlgorithmIdentifier{{SupportedAlgorithms}} OPTIONAL,
  altSignature         BIT STRING OPTIONAL
}
For performance reasons, data transfer APDUs are typically not protected by digital signatures, but by so-called Message Authentication Code (MAC).

A MAC is also called an Integrity Check Value (ICV). This term is used here.
An ICV data type corresponding to SIGNED data type

ICV{ToBeProtected} ::= SEQUENCE {
  toBeProtected  ToBeProtected,
  nonce          OCTET STRING OPTIONAL,
  algorithmIdentifier  AlgorithmIdentifier{{SupportedIcvAlgorithms}},
  icv             OCTET STRING,
  altAlgorithmIdentifier  AlgorithmIdentifier{{SupportedIcvAlgorithms}} OPTIONAL,
  altIcv         OCTET STRING OPTIONAL,
  ...
}

Only necessary if ICV algorithm not negotiated during handshake

Defined in Rec. ITU-T X.510 | ISO/IEC 9594-11
Another approach

AlgorithmIdentifier{ALGORITHM:SupportedAlgorithms} ::= SEQUENCE {
   algorithm ALGORITHM.&id({SupportedAlgorithms}),
   parameters ALGORITHM.&Type({SupportedAlgorithms}{@algorithm}) OPTIONAL,
   ... }

multipleSignatures ALGORITHM ::= {
   PARMS MultipleSignature
   IDENTIFIED BY id-algo-multipleSignatures }

MultipleSignature ::= SEQUENCE OF
   signature SEQUENCE {
      algorithm ALGORITHM.&id({SupportedAlgorithms}),
      parameters ALGORITHM.&Type({SupportedAlgorithms}{@algorithm}) OPTIONAL,
      sign BIT STRING }
Outstanding issue

Two entities need to share symmetric keys
  - For encryption
  - For generation and validation of ICVs

Such keys need to be established in some way

How to migrate current methods are presently not clear!
Methods for establishing symmetric keys between two entities

- Key distribution - not scalable
- Key transport (key encryption) - RSA dependent
- Key agreement, e.g., Diffie-Hellman
  - Results in a shared secret
  - One or more symmetric key generated from shared secret using some algorithm, e.g., HMAC-based Extract-and-Expand Key Derivation Function (HKDF) (RFC 5869)
  - Currently used by IEC 62351-4 and X.510
Methods for establishing symmetric keys between two entities

What are the future symmetric key establishment algorithms?

- Some kind of "Diffie-Hellman" techniques
- Key encapsulation methods, e.g., QC-MDPC (Quasi Cyclic - Moderate Density Parity Check)

Difficult to make a migration strategy without knowing where to go
**Key encapsulation method (KEM)**

- Key encapsulation different from key encryption (using RSA)
- Bob generates a public/private key pair
- Alice feeds Bob's public key into the KEM algorithm to create and encapsulate an AES key
- BOB uses his private key to decapsulate the AES key
- QC-MDPC (Quasi Cyclic - Moderate Density Parity Check) is (hopefully) a quantum-safe KEM
- A hybrid technique increases the security
Specification of extensions with procedures, e.g., for public-key certificates, is in Rec. ITU-T X.509 | ISO/IEC 9594-8 to be finally approved later this year for publication early 2020.


The protocol specified by Rec. ITU-T X.510 | ISO/IEC 9594-11 includes the migration methods (taking its own medicine)
End