CURDLE

Jim Schaad
August Cellars
IETF 96 - Berlin
Private key is a byte string of either 32-bytes or 56-bytes in both cases.
Public key is different for X* and Ed* in the following methods:
1. X* is just the X coordinate while Ed* encodes both the X and the Y coordinates – although only a sign bit for the X coordinate
2. X* computes public by [private]Base while Ed* uses SHA-512 or SHA256 as part of the input process

May want to allow for ECDH methods other than straight ECDH, need to make sure that we understand (and document) what should happen as new ECDH algorithms are adopted (when and if).

Note: known problems with mqv make it unlikely to be used.
EC-MQV is part of Suite B (still true?)
## Curdle PKIX 00

<table>
<thead>
<tr>
<th>Curve</th>
<th>Public Key</th>
<th>Private Key</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>X25519</td>
<td>(id-ecPublicKey or id-ecDH) + id-Curve25519</td>
<td>id-Curve25519</td>
<td>id-ecDH</td>
</tr>
<tr>
<td>X448</td>
<td>(id-ecPublicKey or id-ecDH) + id-Curve448</td>
<td>id-Curve448</td>
<td>id-ecDH</td>
</tr>
<tr>
<td>Ed25519</td>
<td>id-edDSAPublicKey + ed25519 (1)</td>
<td>id-Curve25519</td>
<td>id-edDSASignature</td>
</tr>
<tr>
<td>Ed25519ph</td>
<td>id-edDSAPublicKey + ed25519ph (2)</td>
<td>id-Curve25519ph</td>
<td>id-edDSASignature</td>
</tr>
<tr>
<td>Ed448</td>
<td>id-edDSAPublicKey + ed448 (3)</td>
<td>id-Curve448</td>
<td>id-edDSASignature</td>
</tr>
<tr>
<td>Ed448ph</td>
<td>id-edDSAPublicKey + ed448ph (4)</td>
<td>id-Curve448ph</td>
<td>id-edDSASignature</td>
</tr>
</tbody>
</table>

Nikos – 5/17 – Use OID not enumeration
Since public and private keys are not totally matched up, need to get rid of the generic id-ecPublicKey for signatures

If we want to change things to say that cannot use a single key for multiple algorithms on the ECDH side – can kill id-ecPublicKey in those two rows as well.
## David Benjamin Proposal

<table>
<thead>
<tr>
<th>Curve</th>
<th>Public Key</th>
<th>Private Key</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>X25519</td>
<td>id-Curve25519 or id-Curve25519-HMQV</td>
<td>id-Curve25519 or id-Curve25519-HMQV</td>
<td>id-Curve25519 or id-Curve25519-HMQV</td>
</tr>
<tr>
<td>X448</td>
<td>id-Curve448 or id-Curve448-HMQV</td>
<td>id-Curve448 or id-Curve448-HMQV</td>
<td>id-Curve448 or id-Curve448-HMQV</td>
</tr>
<tr>
<td>Ed25519</td>
<td>id-Ed25519</td>
<td>id-Ed25519</td>
<td>id-Ed25519</td>
</tr>
<tr>
<td>Ed25519ph</td>
<td>id-ed25519ph</td>
<td>id-ed25519ph</td>
<td>id-ed25519ph</td>
</tr>
<tr>
<td>Ed448</td>
<td>id-ed448</td>
<td>id-ed448</td>
<td>id-ed448</td>
</tr>
<tr>
<td>Ed448ph</td>
<td>id-ed448ph</td>
<td>id-ed448ph</td>
<td>id-ed448ph</td>
</tr>
</tbody>
</table>
I am going to ignore the current document as there does not seem to be a great deal of support for it. Based on the ASN.1 encoding, not a great deal to recommend either one. Slightly favor the second approach due to size.

Publishing Capabilities:
- Shorter list if just algorithms
- More likely to affect signatures than ECDH as still need to provide a key
- May assume that algorithms in the list are still ok if list is long enough – shorten to favorites rather than complete support
- Possible to produce a new OID which is just for negotiation if it looks like it really needs to exist.
- LDAP searches for algorithm are more complicated as a larger list of algorithms needs to be done.

Future Algorithms:
- Case of how MQV would be added to the list of algorithms
- Does not work immediately w/o generic curve identifier. Thus does not affect how signature curves are setup.
Curve Misuse:
• Listing algorithms make misuse more blatant. More likely for id-ecPublicKey than edDSA keys. Stupid people can still do stupid things

Memory of somebody saying that knowing just the alg w/o the curve is a problem in some circumstances. – looking at signatures need to go and find the key to know if you can process it. I am not sure if I feel that this is a real problem as you are going to have to spend a lot of time dealing with the certificate before you can use it. Not sure how many instances of – I can’t do this curve – exist in real life.
Makes more sense to do what DH does and say – ECPrivateKey is an OCTET STRING? And just use the OneAsymmetricKey structure as a wrapper (almost the same as PKCS#8)

RFC 5958

Recommendations on sending a public key with the private key? Should we make one.
Discussions
ASN.1

SubjectPublicKeyInfo {PUBLIC-KEY: ISet} ::= SEQUENCE {
  algorithm AlgorithmIdentifier {PUBLIC-KEY, {IOSet}},
  subjectPublicKey BIT STRING }

PrivateKeyInfo ::= SEQUENCE {
  version INTEGER,
  privateKeyAlgorithm AlgorithmIdentifier{PUBLIC-KEY, {...}},
  privateKey OCTET STRING,
  -- Structure of private key is in PUBLIC-KEY.&PrivateKey
  attributes [0] IMPLICIT Attributes OPTIONAL}