Existing ECDHE and ECDSA encodings in TLS 1.3

— **Two problems:** The encodings used in the ECDHE groups `secp256r1`, `secp384r1`, and `secp521r1` and the ECDSA signature algorithms `ecdsa_secp256r1_sha256`, `ecdsa_secp384r1_sha384`, and `ecdsa_secp521r1_sha512` have significant overhead and the ECDSA encoding produces variable-length signatures.

— The document defines new optimal fixed-length encodings and registers new ECDHE groups and ECDSA signature algorithms using these new encodings.

— The encoding are defined as a subset of the bytes in the current encodings. This makes interoperable implementations easy.

— **The new encodings have the same security properties and requirements as the old encodings.**

— The new encodings reduce the size of the ECDHE groups with 33, 49, and 67 bytes and the ECDSA algorithms with an average of 7 bytes.

— When `secp256r1_compact` and `ecdsa_secp256r1_sha256_compact` are used as a replacement for the old encodings they reduce the size of a mutually authenticated TLS handshake with an average 80 bytes.

— These new encodings also work in DTLS 1.3 and are especially useful in cTLS.
  — The alternative to do something cTLS specific seems worse.
  — Many IoT devices want to continue using P-256 and ECDSA.
Compact ECDHE Encoding

Given a UncompressedPointRepresentation structure

```
struct {
    uint8 legacy_form = 4;
    opaque X[coordinate_length];
    opaque Y[coordinate_length];
} UncompressedPointRepresentation;
```

the legacy_form and Y field are omitted to create a CompactRepresentation structure.

```
struct {
    opaque X[coordinate_length];
} CompactRepresentation;
```

For secp256r1 the UncompressedPointRepresentation is 65 bytes and the CompactRepresentation is 32 bytes, saving of 33 bytes.
— The y-coordinate does not affect the shared secret but it is needed for point validation.

— The y-coordinate might also be needed for compatibility with APIs.

— Compact representation have no disadvantages compared to point compression where the sign bit is included.

— To my knowledge, there has never been any patents on compact representation.

---

3.2. Implementation Considerations for Compact Representation

For compatibility with APIs a compressed y-coordinate might be required. For validation or for compatibility with APIs that do not support the full [SECG] format an uncompressed y-coordinate might be required (using the notation in [SECG]):

- If a compressed y-coordinate is required, then the value \(-yp\) set to zero can be used. The compact representation described above can in such a case be transformed into the SECG point compressed format by prepending X with the single byte 0x02 (i.e., \(M = 0x02 \mid X\)).
- If an uncompressed y-coordinate is required, then a y-coordinate has to be calculated following Section 2.3.4 of [SECG] or Appendix C of [RFC6090]. Any of the square roots (see [SECG] or [RFC6090]) can be used. The uncompressed SECG format is \(M = 0x04 \mid X \mid Y\).

For example: The curve P-256 has the parameters (using the notation in [RFC6090])

- \(p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1\)
- \(a = -3\)
- \(b = 410583637251521421293261297804726840911444105993725554835256314039467401291\)

Given an example \(x\):

- \(x = 115792089183334602955468071574470558443443067795108653336398970697772788799766525\)

we can calculate \(y\) as the square root \(w = (x^3 + a \cdot x + b)^{(p + 1)/4} \pmod{p}\)

- \(y = 83438718007019280682007586491862600528145125996401575416632522940595860276856\)

Note that this does not guarantee that \((x, y)\) is on the correct elliptic curve. A full validation according to Section 5.6.2.3.3 of [SP-800-56A] is done by also checking that \(0 \leq x < p\) and that \(y^2 \equiv x^3 + a \cdot x + b \pmod{p}\). The implementation MUST perform public-key validation.
Compact ECDSA Encoding

Given a variable-length DER-encoded ECDSA-Sig-Value structure

the SEQUENCE type, INTEGER type, and length fields are omitted and if necessary, the two INTEGER value fields are truncated (at most a single zero byte) or left padded with zeroes to the fixed length L.

For secp256r1 the ecdsa_secp256r1_sha256 example is 71 bytes and the ecdsa_secp256r1_sha256_compact signature is 64 bytes, saving 7 bytes.
New IANA Registrations

— Three new ECDHE groups for P-256, P-384, and P-521

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Recommended</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD1</td>
<td>secp256r1_compact</td>
<td>Y</td>
<td>[This-Document]</td>
</tr>
<tr>
<td>TBD2</td>
<td>secp384r1_compact</td>
<td>Y</td>
<td>[This-Document]</td>
</tr>
<tr>
<td>TBD3</td>
<td>secp521r1_compact</td>
<td>Y</td>
<td>[This-Document]</td>
</tr>
</tbody>
</table>

Table 1: Compact ECDHE Groups

— Three new signature algorithms for P-256, P-384, and P-521

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Recommended</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD4</td>
<td>ecdsa_secp256r1_sha256_compact</td>
<td>Y</td>
<td>[This-Document]</td>
</tr>
<tr>
<td>TBD5</td>
<td>ecdsa_secp384r1_sha384_compact</td>
<td>Y</td>
<td>[This-Document]</td>
</tr>
<tr>
<td>TBD6</td>
<td>ecdsa_secp521r1_sha512_compact</td>
<td>Y</td>
<td>[This-Document]</td>
</tr>
</tbody>
</table>

Table 2: Compact ECDSA Signature Algorithms