SOS (Simple Octet String) [representing new ECC in OpenPGP]

NIIBE Yutaka

OpenPGP / IETF 110

2021-03-11

SOS is a variant for MPI

- OpenPGP defines a Multi-Precision Integer format…
- …but ships some non-MPI things in that format (ECC points in particular).
- SOS formalizes this and makes it easier to work with, for new curves like `curve448` in particular.

SOS Definition

> An SOS consists of two pieces: a two-octet scalar that is the length of the SOS in bits followed by an opaque string of octets

- Two interpretations:
  - 8× length (in octets) of opaque octet string
  - Length (in bits) of a big-endian number
- Backward-compatible with OpenPGP's MPI

Classic ECC in OpenPGP with SOS

- Existing specification for classic ECC can just replace the word "MPI" with "SOS".
- Cleaner definition than RFC 6637, interoperable.

Classic ECC in OpenPGP with SOS

Before:

The point is encoded in the Multiprecision Integer (MPI) format [RFC4880]. The content of the MPI is the following:

B = 04 || x || y

where x and y are coordinates of the point P = (x, y), each encoded in the big-endian format and zero-padded to the adjusted underlying field size. The adjusted underlying field size is the underlying field size that is rounded up to the nearest 8-bit boundary.

After:

(When an OID is one of NIST Curves,) See RFC 8422 (Section 5.4.1. Uncompressed Point Format for NIST Curves) for its semantics.

Modern ECC in OpenPGP with SOS

- For new modern curves, use SOS to represent:
  - Scalars
  - EC points
  - In native format of underlying algorithm

Already-deployed Modern ECC

- Ed25519 key and signature predates SOS
- Curve25519 key and encryption by ECDH predates both SOS and X25519

We need special care for these algorithms.

Special care: EdDSA with Ed25519

- Keys use prefix 0x40 to represent EC point
- But no 0x40 prefix for EC point in Signature part of R
- Preceding zero-removal:
  - Not an issue for keys, thanks to 0x40 prefix
  - Still an issue for signature parts of R and S
- For interop, SOS-aware implementations must still handle prefix and zero-removal for Ed25519 as a special case.

Special care: ECDH with Curve25519

- Keys use prefix 0x40 to represent an EC point.
- Secret key is scalar represented as big-endian MPI
- Formal spec X25519 has no prefix, uses little-endian representation for secret scalar.
- For interop, SOS-aware implementations must still use prefix, and secret scalar must be big-endian MPI for ECDH with Curve25519 as a special case.

Why do zero-removal at all?

- Historical artifact? (ASN.1 BER and DER both require it)
- SOS delegates that choice to the underlying crypto algorithm, **not** the OpenPGP layer.

SOS Principle

- SOS uses OpenPGP to **convey** data of underlying crypto algorithm.
- Underlying algorithm defines the data format natively.

Alternatives 1/3: “easiest”

- Mimic what we did for Ed25519 (prefix 0x40 for key, not for signature, and zero-removal/zero-recovery)
- Mimic Curve25519 (prefix 0x40 for key, big-endian secret scalar)

Pros

- Easier for existing implementations

Cons

- More code complexity for new curves
- Translation required for crypto libraries

Alternatives 2/3: “per-curve”

- Define a specific data format for each curve

Pros

- Simple to dispatch to crypto library
- Easier for new implementations

Cons

- Impossible to know how to skip over unknown curve data

Alternatives 3/3: “JOS”

- Define simpler opaque Octet String (length in octets, not bits): “Just an Octet String”

Pros

- Simpler, if we were writing OpenPGP from scratch

Cons

- Would require new pubkey algorithm numbers for EdDSA-JOS and ECDH-JOS

My conclusion

- SOS is a compromise to introduce other modern curves of ECC without diverging from standard implementations.
- Looks a bit strange, but backward-compatibility is good.
- Recommend adoption by the WG.

Questions?