After reading version 14 of the BGPSec protocol draft and after discussing the update between us, Michael Baer (BIRD implementer) and I (Quagga Implementer) want to propose some changes for generation of the “Sequence of Octets to be Signed” (SOS) in the draft on pg. 15. This change would modify the order of information within the SOS as well as the order of attributes within the “Secure_Path” Segment listed on pg. 8.

NONE of the changes has any impact on the information that is put on the wire in regards to adding or removing data. The only on the wire change is the ordering of the attributes within the Secure_Path Segment.

As we are all aware, the most expensive operation within the BGPSEC protocol is the crypto operation, especially the Path verification.
With the proposed modification of the SOS, implementers will be able to utilize more efficient and higher performing software mechanisms to validate the complete chain of signatures in an update. The current form makes this more difficult.

Our request does remove some data from the previous SOS structure, changes the order of the remaining attributes within the SOS and includes the re-ordering of one data segment on the wire, which will facilitate the SOS generation.

1) Request for re-ordering the Secure_Path segment
The first request deals with modifying the order of the Secure_Path segment. This modification will become more obvious later on when we explain our request for changes in the structure of “Sequence of Octets to be Signed” (SOS) on pg. 15.
This is also the only change that has an affect on the data on the “wire” but again only regarding the order itself, NOT the content.

The current format as it is shown on pg. 8 is as follows:

```
+-------------------------------+
| AS Number  (4 octets)         |
+-------------------------------+
| pCount     (1 octet)          |
+-------------------------------+
| Flags      (1 octet)          |
+-------------------------------+
```

We request to move the “AS Number” field to the end of the signature segment. This results in the following structure:

```
+-------------------------------+
| pCount     (1 octet)          |
+-------------------------------+
| Flags      (1 octet)          |
+-------------------------------+
| AS Number  (4 octets)         |
+-------------------------------+
```

The reason for this minor change becomes more clear when we explain our request for modifying the SOS structure. But as a little preview for where we want to go with this, consider the following:

Having a set of Secure_Path segments, the last field of the following segment equals the “Target AS” needed in the SOS structure. But this becomes more obvious later on.

2) Modifying the SOS structure”

The current structure as it is presented on pg. 15 of draft-14 is as follows:

```
+--------------------------------+
| Target AS Number               |
+--------------------------------+
| AS Number                      |
+--------------------------------+
| pCount                         |
+--------------------------------+
| Flags                          |
+--------------------------------+
| Previous Secure_Path           |
+--------------------------------+
| Previous Signature_Block       |
+--------------------------------+
| AFI                            |
+--------------------------------+
| SAFI                           |
+--------------------------------+
| NLRI                           |
+--------------------------------+
```

This structure is very inefficient for signature validation because for each signature validation the structure needs to be newly regenerated.

One major change in version-14 compared to the preceding drafts is the inclusion of all previous signatures to the SOS structure. In the previous draft only the directly preceding signature was part of the SOS. Version-14 introduces an additional overhead of approximately 91-93 bytes (69-71 for signature + 20 SKI + 2 signature length) or ~92 extra bytes per Signature.

This means that verifying a 10 hop path, the following additional overhead for signatures must be added to each SOS in comparison to draft 13:

(assumed 92 bytes on average per signature)

```
SOS overhead 10 signatures: +828 bytes
SOS overhead  9 signatures: +736 bytes
SOS overhead  8 signatures: +644 bytes
SOS overhead  7 signatures: +552 bytes
SOS overhead  6 signatures: +460 bytes
SOS overhead  5 signatures: +368 bytes
SOS overhead  4 signatures: +276 bytes
SOS overhead  3 signatures: +184 bytes
SOS overhead  2 signatures: + 92 bytes
```

For sequential verification only the maximum memory overhead comes into place because each consecutive verification will have an SOS size less then the previous one.
For parallel verification though each verification itself requires the necessary memory overhead to the SOS which will end up with:

Cumulative overhead for a 10 hop path: 4,140 bytes

And this is only the additional memory consumption added with the signatures. On top of that comes the prefix information {AFI, SAFI, NLRI}   -> (5...21 bytes), Path information {AS, pCount, Flags} -> 6 bytes each, and 1 byte for algo ID.

Depending where the data is located during the hash generation (e.g. L2 cache) the additional memory accesses could further hinder performance and negatively affect convergence time.

Furthermore, the newly proposed (version-14 draft) SOS includes Secure_Path Length and Signature_Block Length, both of which are overwritten at each hop. This imposes the additional burden of regenerating these length fields for the SOS corresponding to each signature verification. This again means that each parallel working thread is required to generate its own SOS for signature validation (see earlier discussion). Hence, it is not desirable to include these length fields in the SOS at the sender. Removing these will not create a security risk.

The idea is to generate an SOS that can be re-used so that it only has to be generated once and then can be utilized without any modification for all signature verifications within an update – regardless if sequential or parallel processing is used.

The proposed modification will result in the following SOS structure:
For simplification we combine the signature and path segments shown below into a combined segment (in the SOS):

```
+-----------------------------+
| SKI                  (n-1) |\
+-----------------------------+ \
| Signarture Length    (n-1) |--- Signature Segment   (n-1)
+-----------------------------+ /
| Signature            (n-1) |/
+-----------------------------+
| pCount               (n)   |\
+-----------------------------+ \
| Flags                (n)   |--- Secure_Path Segment (n)
+-----------------------------+ /
| AS Number            (n)   |/
+-----------------------------+
```

The simplification in imploded form looks as follows:

```
+-----------------------------+
| Signature Segment    (n-1) |
+-----------------------------+
| Secure_Path Segment  (n)    |
+-----------------------------+
```

Proposed SOS Structure: (See example on next page for n=3)

```
+----------------------------------------+
| Target AS Number                       |
+----------------------------------------+\
| Signature Segment          (n-1)       | \
+----------------------------------------+ |
| Secure_Path Segment        (n)         | |
+----------------------------------------+ \
...                                         > For n Hops
+----------------------------------------+ /
| Signature Segment          (1 origin) | |
+----------------------------------------+ |
| Secure_Path Segment        (2)         | /
+----------------------------------------+/
| Secure_Path Segment        (1 origin) |
+----------------------------------------+
| Algorithm Suite Identifier             |
+----------------------------------------+
| AFI                                    |
+----------------------------------------+
| SAFI                                   |
+----------------------------------------+
| NLRI                                   |
+----------------------------------------+
```

This structure allows the generation of one single SOS that can be accessed simultaneously by multiple threads (one for each signature verification).

With this structure an update containing 10 Signatures contains the same overhead of +828 bytes but here it does not need to be re-generated for each signature validation. Independent of whether the validation is performed sequential or parallel, the overhead remains the same and will NOT grow to an extra 4,140 bytes as outlined earlier. This will result in a net saving of +3,312 bytes for a path with 10 signatures in addition to the time saved generating the SOS for each validation separately.

Example for generation and processing the new proposed SOS structure for a signed path from AS1 to AS4:
AS1—AS2—AS3-AS4

```
           +----------------------+
SOS 3----->| AS 4                 |   <- (Target AS for signature 3)
         | +----------------------+
         | | Signature_Segment (2)|
         | +----------------------+
         | | pCount            (3)| \
         | +----------------------+  \
         | | Flags             (3)| --- Secure_Path Segment (3)
         | +----------------------+  /
SOS 2----+>| AS 3              (3)| / <- (Target AS for signature 2)
       | | +----------------------+
       | | | Signature_Segment (1)|
       | | +----------------------+
       | | | pCount            (2)| \
       | | +----------------------+  \
       | | | Flags             (2)| --- Secure_Path Segment (2)
       | | +----------------------+  /
SOS 1--+-+>| AS 2              (2)| / <- (Target AS for signature 1)
     | | | +----------------------+
     | | | | pCount            (1)| \
     | | | +----------------------+  \
     | | | | Flags             (1)| --- Secure_Path Segment (origin)
     | | | +----------------------+  /
     | | | | AS 1              (1)| /
     | | | +----------------------+
     | | | | Algorithm Suite ID   |
     | | | +----------------------+
     | | | | AFI                  |
     | | | +----------------------+
     | | | | SAFI                 |
     | | | +----------------------+
     | | | | NLRI                 |
END  +-+-+>+----------------------+
```

As one can clearly observe the receiver needs only to generate one single SOS and can utilize it for validation of all previous signatures without the need to regenerate the SOS at each step.

Better even, the new SOS allows:
- sequential validation processing without the need to regenerate the SOS data for each validation process; just use pointer arithmetic to specify start of the structure
- parallel validation processing using the same memory location.

Thanks,

Oliver Borchert (NIST) & Michael Baer (PARSONS)