BPSec COSE Contexts

BRIAN SIPOS

RKF ENGINEERING SOLUTIONS

IETF109

Motivations for COSE/BPSec

BPSec security contexts are tailored to specific situations and optimized for minimum-encoded-size security blocks.

- Original BPSec focus is on symmetric-keyed algorithms.

For internet-facing nodes, possibly as subnetwork gateways, there is a need for PKI-integrated security.

- This was indicated also by SECDIR review of BPSec draft.

Don’t want to reinvent the wheel, and CBOR Object Signing and Encryption (COSE) already provides syntax and semantics for current and future security algorithms.

Goals for COSE/BPSec

No not alter BPSec structures or requirements.

- This is purely an extension within the existing security context mechanism.

Handle current symmetric-keyed and PKIX algorithms.

- Leverage existing algorithm definitions.

Follow algorithm-use and key-use best practices.

- Avoid key overuse, use random content encryption keys.

Inherit future gains made by COSE off-the-shelf algorithms.

Proposed Security Contexts

One context codepoint with result types defined for each BPSec block type:

- COSE Integrity results (MAC and Signature)
- COSE Confidentiality results (AEAD Encrypt)

Public keys in context parameters to de-duplicate data.

- Potential future extensions could provide additional supporting data (e.g. OCSP stapling).

Full COSE messages in each target's result.

- Reuse COSE message tags as result type codes.
- Allows an application to use any current or future COSE algorithm types (and combinations).
- Allows multiple recipients for a single security block (both BIB and BCB).
- Interoperability requirements are defined in a COSE Profile (next slide).

Proposed COSE Profile

Required algorithms for AES-GCM-256, AES key-wrap, and HMAC-SHA2-256.

- AES key-wrap was added to required list since last draft.

Recommended algorithms for EC and RSA signing and key-wrap/key-generation.

- Additional public key material can be included as security parameters, applying to all results in the block.

BPSec Block	COSE Layer	Name	Code	Implementation Requirements
Integrity	1	HMAC 256/256	5	Required
Integrity	1	ES256	-7	Recommended
Integrity	1	EdDSA	-8	Recommended
Integrity	1	PS256	-37	Recommended
Confidentiality	1	A256GCM	3	Required
Integrity or Confidentiality	2	A256KW	-5	Required
Integrity or Confidentiality	2	ECDH-ES + A256KW	-31	Recommended
Integrity or Confidentiality	2	RSAES-OAEP w/ SHA-256	-41	Recommended

Table 4: Interoperability Algorithms

Specializations to BPSec Requirements

The COSE contexts require AEAD encryption and require that both BIB and BCB include the primary block and target block metadata as AAD.

- This binds the security result to that exact block and its containing bundle.

This means that AAD cannot change after BIB or BCP is applied.

- The primary block is required to be immutable already, so no issue there.

The ASB security parameters are still *not* included as AAD.

- This is functionally equivalent to the COSE “protected header” vs. “unprotected header” separation.
- Security parameters are necessary for signature verification but tampering with them will just cause verification to fail.

Desired WG Direction

This is not intended to replace or supersede existing BPSec interoperability contexts (`draft-ietf-dtn-bpsec-interop-sc`).

The point here is to allow BPSec in a PKIX environment in the very near term.

- COSE is a known quantity with existing coding and processing tools.
- Validation of a Node ID within a PKIX certificate are already defined in TCPCLv4.

Some secondary questions remain:

- E.g. how does a security acceptor handle a BIB signed by a key with a certificate for a different Node ID than the security source? Base BPSec doesn't really deal with identity logic.
- A BIB with an "x5t" reference can include the signing certificate (chain). Should a BCB with an "x5t" recipient also include the recipient certificate itself?
- Similar to the question for TCPCLv4, is there a need for a DTN-specific Extended Key Usage OID?

No real review has been done on this draft yet!