Group OSCORE - Secure Group Communication for CoAP

draft-ietf-core-oscore-groupcomm-12

Marco Tiloca, RISE
Göran Selander, Ericsson
Francesca Palombini, Ericsson
John Mattsson, Ericsson
Jiye Park, Universität Duisburg-Essen

IETF 111, CoRE WG, July 28th, 2021

Update since the March meeting

- Version -12 submitted

- Main updates
  - Recycling of Group IDs in the group (Christian)
  - Security of using an identity public key for both signing and Diffie-Hellman (Ben [1][2])
  - Major changes to the message processing, especially in group mode (John)
  - Clarified security properties

[1] https://mailarchive.ietf.org/arch/msg/core/ujj_I-LlqW9fq__quh-YqKS0fF0/

[2] https://mailarchive.ietf.org/arch/msg/core/YRNXvtiFmHLk5YkXK8-uJg-t3NU/

Updates from -12

- **Recycling of Group IDs in a group – Reminder: the Group ID changes when rekeying**
  - It used to be forbidden, to prevent a notification from matching two long-lived observations
- The Group Manager (GM) retains the Gid obtained by a node when joining, i.e. its “Birth Gid”
  - Before rekeying the group, the GM checks if the new Gid is any current member’s “Birth Gid”
  - If such members are found, the GM removes them from the group and rekeys accordingly
- Those evicted nodes will ask the GM for the latest keying material
  - Since they are not group members anymore, they receive error responses
  - Eventually, they will re-join the group and thus terminate their observations
- If any of those nodes re-joins before another rekeying has happened
  - The Group Manager MUST NOT rekey the group again upon its joining
- Recycling Group IDs is safe → A group can live forever

Updates from -12

- **Security of using an identity key for both signing and Diffie-Hellman** [3][4] – #72 #73
  - If signing keys use Ed25519 (Ed448) they are converted to Curve25519 (Curve448) for DH
  - The computed DH secret is used to generate encryption keys for the pairwise mode
  - Both uses have the same goal and policy: group communication under a Security Context

- This double-use deviates from common best practices → Security has to be well proven
  - Build on and extend the proof at [5], as focused on (but not limited to) ECIES settings
  - Now proven to be secure specifically in Group OSCORE, see [6] – Thanks to Erik Thormarker!

[3] https://mailarchive.ietf.org/arch/msg/core/ujj_I-LlqW9fq__quh-YqKS0fF0/
[4] https://mailarchive.ietf.org/arch/msg/core/YRNXvtiFmHLk5YkXK8-uJg-t3NU/
[5] https://eprint.iacr.org/2011/615.pdf
[6] https://eprint.iacr.org/2021/509.pdf

Updates from -12

- **Security of using an identity key for both signing and Diffie-Hellman**
  - Adapted derivation of pairwise keys, explicitly involving the two peers' public keys (see Section 2.4.1)

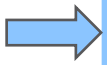
```
Pairwise Sender Key    = HKDF(Sender Key, IKM-Sender, info, L)
Pairwise Recipient Key = HKDF(Recipient Key, IKM-Recipient, info, L)

with

IKM-Sender    = Sender Pub Key | Recipient Pub Key | Shared Secret
IKM-Recipient = Recipient Pub Key | Sender Pub Key | Shared Secret
```

- (Cryptographic) security considerations supporting the procedure's correctness
  - For ECDSA signing keys, [6] proves that the proof from [5] is applicable also to Group OSCORE
  - For EdDSA signing keys, [6] builds a new proof from the one in [5], specific to Group OSCORE
  - Dedicated discussion on converting from Ed25519 (Ed448) to Curve25519 (Curve448)

[5] https://eprint.iacr.org/2011/615.pdf
[6] https://eprint.iacr.org/2021/509.pdf

Updates from -12

- **Admitted formats of public keys (see Section 2.3)**
  - Must provide the full set of information about the public key algorithm
  - Relevant examples:
    - CWT, see *RFC8392*
    - Unprotected CWT claim set, see *draft-ietf-rats-uccs*
    - X.509 certificates, see *RFC7925*
    - C509 certificates, see *draft-ietf-cose-cbor-encoded-cert*



```
{ /CWT claims list/
  2: "42-50-31-FF-EF-37-32-39", /sub/
  8: { /cnf/
       1: { /COSE_Key/
       1: 1,
      -1: 4, /X25519/
      -2: h'b1a3e89460e88d3a8d54211dc95f0b
            903ff205eb71912d6db8f4af980d2db83a',
          }
  }
}
```

- All public keys used in the group
  - Must have the same single format used in the group
  - Must be compatible with the algorithm and related parameters used in the group

Updates from -12

- **Now an OSCORE group can work in three ways**
  - "Group mode" only; "Pairwise mode" only (NEW); both modes
  - Group members know what is used in the group – Learned when joining, or at early group discovery
- **Consistent revision of the Security Context – Additions to OSCORE (RFC8613)**
  - (*) Specific to the group mode
  - (^) Specific to the pairwise mode
  - Some recently added parameters
- **In group mode**
  - Encrypt with Signature Encryption Algorithm
  - Sign with Signature Algorithm
  - 1 common Group Encryption Key derived like a Sender/Recipient Key, through HKDF()
- **In pairwise mode**
  - Encrypt with AEAD Algorithm (RFC 8613)
  - Derive pairwise keys with Pairwise Key Agreement Algorithm

Context Component	New Information Elements
Common Context	Group Manager Public Key * Signature Encryption Algorithm * Signature Algorithm * Group Encryption Key ^ Pairwise Key Agreement Algorithm
Sender Context	Endpoint's own public and private key pair ^ Pairwise Sender Keys for the other endpoints
Each Recipient Context	Public key of the other endpoint ^ Pairwise Recipient Key of the other endpoint

Figure 1: Additions to the OSCORE Security Context. The optional elements labeled with * (with ^) are present only if the group uses the group mode (the pairwise mode).

Updates from -12

- **Now an OSCORE group can work in three ways**
  - "Group mode" only; "Pairwise mode" only (NEW); both modes
  - Group members know what is used in the group – Learned when joining, or at early group discovery

- **Consistent revision of the External AAD**
  - Some recently added parameters

- **Removed some parameters**
  - Capabilities and properties of algorithms
  - Now embedded in the public keys

```
external_aad = bstr .cbor aad_array

aad_array = [
   oscore_version : uint,
   algorithms : [alg_aead : int / tstr / null,
                 alg_signature_enc : int / tstr / null,
                 alg_signature : int / tstr / null,
                 alg_pairwise_key_agreement : int / tstr / null],
   request_kid : bstr,
   request_piv : bstr,
   options : bstr,
   request_kid_context : bstr,
   OSCORE_option: bstr,
   sender_public_key: bstr,
   gm_public_key: bstr / null
]
```

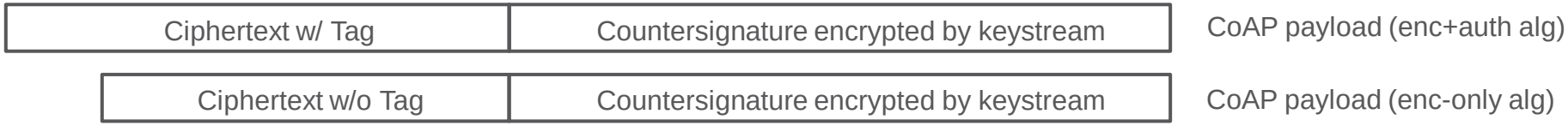
- **Added Group Manager public key**
  - More accurate "group description" covered by the external_aad
  - Prevents a "group cloning attack", discussed in the security considerations

Updates from -12

- **In group mode, the countersignature is separately encrypted**
  - Encrypt COSE plaintext → Sign the COSE ciphertext → Encrypt the countersignature separately
  - Group privacy: not possible to track a user across two groups, unless the tracker is a member of both
- **Countersignature encryption**
  - ENC_COUNTERSIGNATURE = SIGNATURE XOR KEYSTREAM
  - KEYSTREAM derived on a per-message basis from the new Group Encryption Key
- **Keystream derivation**
  - KEYSTREAM = HKDF(salt, IKM, info, length)
    - salt : Partial IV used to protect the message
    - IKM : Group Encryption Key
    - info : [Sender ID of the Partial IV generator; Group ID; True/False (req/resp); length]
    - length : Same length as the countersignature
- **Receiver side: verify the countersignature first, then decrypt the COSE ciphertext**

Updates from -12

- **In group mode, admit also encryption-only algorithms**
  - Expected registration as COSE algorithms
  - Some applications may not desire/afford both a signature and an integrity Tag in each message
  - Source authentication still ensured, thanks to the signature and what is covered by the external_aad



- **Even with encryption-only algorithms, i.e. without integrity Tag, we still want to …**
  - Admit external signature checkers and always verify group membership when receiving a message
  - But a group member might leave, become external signature checker and inject re-signed messages
  - Other nodes would believe the sender to still be a group member – They still have its public key!

Therefore …

Updates from -12

- **Stricter management of group keying material, including group rekeying**
  - If a node leaves, the GM must rekey the group
  - Rekeying messages must specify the “stale Sender IDs” of the current key generation
    - Sender IDs earlier relinquished due to a change requested by the owning node; or
    - Sender IDs belonging to a leaving node
- **The rekeyed group members obtain the stale Sender IDs**
  - They delete the associated public keys used in the group
  - They delete the associated Recipient Contexts in the group

  → They rely on their owned public keys to assert the group membership of sender endpoints
- **A group member may miss a group rekeying instance**
  - The GM allows to retrieve an aggregate set of stale Sender IDs for the most recent key generations
- **Detailed mechanics for achieving this are out of scope for Group OSCORE**
  - The GM defined in *draft-ietf-ace-key-groupcomm-oscore* provides these functionalities

Updates from -12

- **New Section 8.5 on external signature checkers**
  - Guidelines on message processing and corner cases

- **Removed three Appendices**
  - “No Verification of Signatures in Group Mode” – Can’t be considered secure
  - “Example Values with COSE Capabilities” – Moot (see admitted formats of public keys)
  - “Parameter Extensibility for Future COSE Algorithms” – Moot (see admitted formats of public keys)

- **Extended security considerations**
  - Revised considerations on the group mode, emphasizing its security properties
  - New Section 10.8 about the prevented “group cloning attack”
  - More cryptographic considerations, on using the same key pair for signing and for Diffie-Hellman

Next steps

- Improve security considerations
  - Mostly on the group mode, based on the latest changes
- Double-check that the few open Github issues are well addressed
- Submit v -13
  - If no further issues arise, it should be ready for a 2nd WGLC

Thank you!

Comments/questions?

https://github.com/core-wg/oscore-groupcomm