Stream: Internet Engineering Task Force (IETF)
RFC: 9921
Category: Standards Track
Published: February 2026
ISSN: 2070-1721
Authors: H. Birkholz, *Fraunhofer SIT*; T. Fossati, *Linaro*; M. Riechert, *Microsoft*

RFC 9921
CBOR Object Signing and Encryption (COSE) Header Parameter for Timestamp Tokens as Defined in RFC 3161

Abstract

This document defines two CBOR Object Signing and Encryption (COSE) header parameters for incorporating timestamping based on RFC 3161 into COSE message structures (`COSE_Sign` and `COSE_Sign1`). This enables the use of established timestamping infrastructure per RFC 3161 in COSE-based protocols.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9921.

Copyright Notice

Table of Contents

1. Introduction

RFC 3161 [RFC3161] provides a method for timestamping a message digest to prove that it was created before a given time.

This document defines two new CBOR Object Signing and Encryption (COSE) [RFC9052] header parameters that carry the `TimeStampToken` (TST) output [RFC3161], thus allowing existing and widely deployed trust infrastructure to be used with COSE structures used for signing (`COSE_Sign` and `COSE_Sign1`).

1.1. Use Cases

This section discusses two use cases, each representing one of the two modes of use defined in Section 2. As the security characteristics of the two cases differ, care must be taken when choosing the appropriate mode for a given application. See Section 5.1 for a discussion on the security of the implementations.

The primary use case is that of "long-term signatures", i.e., signatures that can still be verified even after the signing certificate has expired. This can address situations where it is important to prevent subsequent denial by the signer or to verify signatures made using (very) short-term certificates. To achieve this, the document signer acquires a fresh TST for the document's signature from a trusted Time Stamping Authority (TSA) [RFC3161] and concatenates it with the document. Later, when a relying party verifies the signed document and its associated TST, they can be certain that the document was signed *at least* at the time specified by the TSA and that the signing certificate was valid at the time the signature was made.

This primary usage scenario motivates the "COSE, Then Timestamp" mode described in Section 2.1.

The second use case is new. It is the notarization of a signed document by registering it with a transparency service. This is common practice for ensuring the accountability and auditability of issued documents, which are typically referred to as "statements" in this context. It is also common practice to only register the signed parts of a statement (the "signed statement" portion) with a transparency service, in order to reduce the complexity of consistency checks at a later stage and to avoid the need to retrieve or reconstruct unsigned parts. Once the signed parts of a document have been registered in the append-only log at a transparency service, the log entry cannot be changed. In order to avoid losing the TST during the registration process, the TST must be included in the signed statement. To achieve this, the issuer acquires a TST from a TSA, includes it in the to-be-signed part of the statement so that the resulting signed statement includes the TST, and then registers the signed parts (rendering it a "transparent statement"). Later on, a relying party consuming the transparent statement including the TST can be certain that the statement was signed by the issuer *at least* at the time specified by the TSA. If the issuer's signing key has expired (or has been compromised), the authenticity of the statement can be ascertained by ensuring that no revocation information was made public before the time asserted by the issuer and registered at the transparency service.

This new usage scenario motivates the "Timestamp, Then COSE" mode defined in Section 2.2.

1.2. Requirements Notation

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Modes of Use

There are two different modes of composing COSE protection and timestamping, motivated by the usage scenarios discussed above.

The diagrams in this section illustrate the processing flow of the specified modes. For simplicity, only the `COSE_Sign1` processing is shown. Similar diagrams for `COSE_Sign` can be derived by allowing multiple `private-key` parallelogram boxes and replacing the label `[signature]` with `[signatures]`.

2.1. COSE, Then Timestamp (CTT)

Figure 1 shows the case where the signature(s) field of the COSE Signed Message is digested and submitted to a TSA to be timestamped. The obtained timestamp token is then added back as an unprotected header into the same COSE object.

This mode is utilized when a record of the timing of the signature operation is desired.

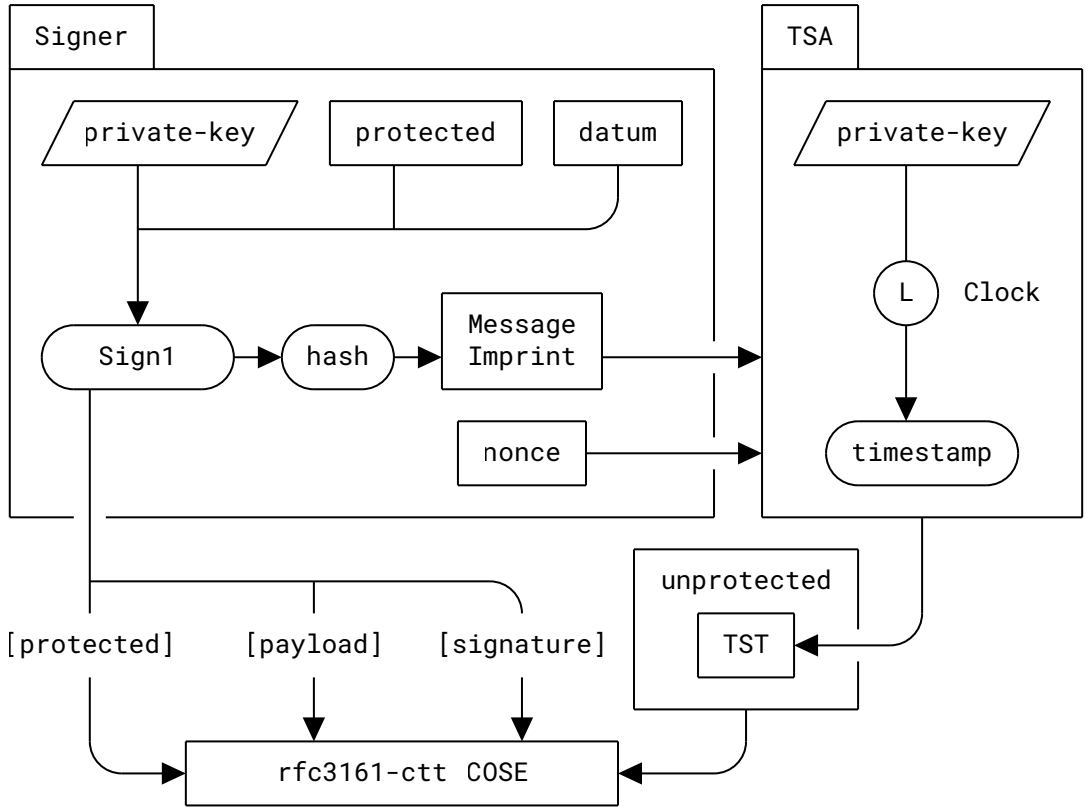

Figure 1: COSE, Then Timestamp (CTT)

In this context, timestamp tokens are similar to a countersignature made by the TSA.

2.2. Timestamp, Then COSE (TTC)

Figure 2 shows the case where a datum is first digested and submitted to a TSA to be timestamped.

This mode is used to wrap the signed document and its timestamp together in an immutable payload.

A COSE Signed Message is then built as follows:

- The obtained timestamp token is added to the protected headers.
- The original datum becomes the payload of the COSE Signed Message.

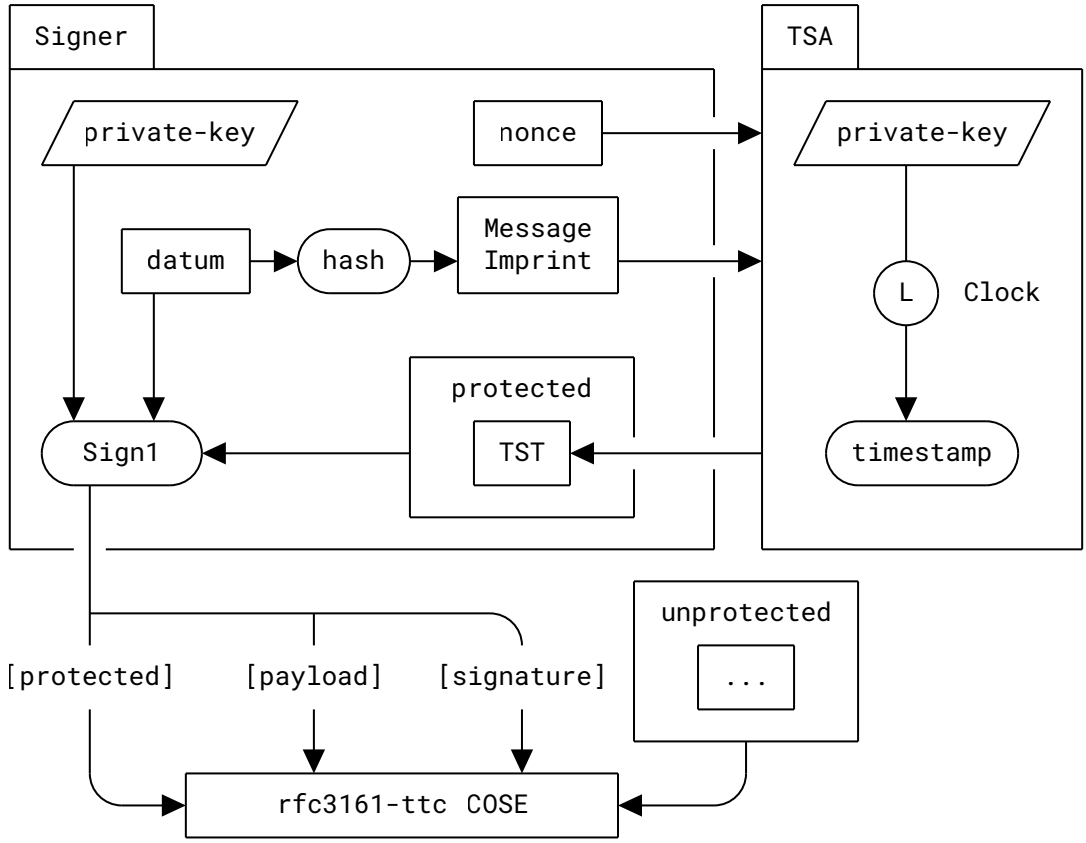

Figure 2: Timestamp, Then COSE (TTC)

3. Timestamp Tokens per RFC 3161: COSE Header Parameters

The two modes described in Sections 2.2 and 2.1 use different inputs into the timestamping machinery and consequently create different kinds of bindings between COSE and TST. To clearly separate their semantics, two different COSE header parameters are defined as described in the following subsections.

3.1. `3161-ctt`

The `3161-ctt` COSE *unprotected* header parameter **MUST** be used for the mode described in Section 2.1.

The `3161-ctt` unprotected header parameter contains a DER-encoded TST [RFC3161] wrapped in a CBOR byte string (Major type 2).

The `MessageImprint` sent in the request to the TSA **MUST** be

- the hash of the CBOR-encoded signature field of the `COSE_Sign1` message, or

- the hash of the CBOR-encoded signatures field of the `COSE_Sign` message.

In either case, to minimize dependencies, the hash algorithm **SHOULD** be the same as the algorithm used for signing the COSE message. This may not be possible if the timestamp token has been obtained outside the processing context in which the COSE object is assembled.

Refer to Sections 3.1.1 and 3.1.2 for concrete examples of `MessageImprint` computation.

3.1.1. `MessageImprint` Computation for `COSE_Sign1`

The following illustrates how `MessageImprint` is computed using a sample `COSE_Sign1` message.

Given the `COSE_Sign1` message

```
18(
  [
    / protected h'a10126' / << {
       / alg / 1:-7 / ECDSA 256 /
      } >>,
    / unprotected / {
      / kid / 4:'11'
    },
    / payload / 'This is the content.',
    / signature / h'8eb33e4ca31d1c465ab05aac34cc6b23d58fef5c083106c4
d25a91aef0b0117e2af9a291aa32e14ab834dc56ed2a223444547e01f11d3b0916e5
a4c345cacb36'
  ]
)
```

the `bstr`-wrapped `signature`

```
58 40                                           # bytes(64)
   8eb33e4ca31d1c465ab05aac34cc6b23
   d58fef5c083106c4d25a91aef0b0117e
   2af9a291aa32e14ab834dc56ed2a2234
   44547e01f11d3b0916e5a4c345cacb36
```

(including the heading bytes `0x5840`) is used as input for computing the `MessageImprint`.

When using SHA-256, the resulting `MessageImprint` is

```
SEQUENCE {
  SEQUENCE {
    OBJECT IDENTIFIER sha-256 (2 16 840 1 101 3 4 2 1)
    NULL
    }
  OCTET STRING
    44 C2 41 9D 13 1D 53 D5 55 84 B5 DD 33 B7 88 C2
    4E 55 1C 6D 44 B1 AF C8 B2 B8 5E 69 54 76 3B 4E
  }
```

3.1.2. `MessageImprint` Computation for `COSE_Sign`

The following illustrates how `MessageImprint` is computed using a sample `COSE_Sign` message.

Given the `COSE_Sign` message

```
98(
  [
    / protected / h'',
    / unprotected / {},
    / payload / 'This is the content.',
    / signatures / [
      [
        / protected h'a10126' / << {
            / alg / 1:-7 / ECDSA 256 /
          } >>,
        / unprotected / {
          / kid / 4:'11'
        },
        / signature / h'e2aeafd40d69d19dfe6e52077c5d7ff4e408282cbefb
5d06cbf414af2e19d982ac45ac98b8544c908b4507de1e90b717c3d34816fe926a2b
98f53afd2fa0f30a'
      ]
    ]
  ]
)
```

the `signatures` array

```
81                                      # array(1)
   83                                   # array(3)
      43                                # bytes(3)
         a10126
      a1                                # map(1)
         04                             # unsigned(4)
         42                             # bytes(2)
            3131                        # "11"
      58 40                             # bytes(64)
         e2aeafd40d69d19dfe6e52077c5d7ff4
         e408282cbefb5d06cbf414af2e19d982
         ac45ac98b8544c908b4507de1e90b717
         c3d34816fe926a2b98f53afd2fa0f30a
```

is used as input for computing the `MessageImprint`.

When using SHA-256, the resulting `MessageImprint` is

```
SEQUENCE {
  SEQUENCE {
    OBJECT IDENTIFIER sha-256 (2 16 840 1 101 3 4 2 1)
    NULL
    }
  OCTET STRING
    80 3F AD A2 91 2D 6B 7A 83 3A 27 BD 96 1C C0 5B
    C1 CC 16 47 59 B1 C5 6F 7A A7 71 E4 E2 15 26 F7
  }
```

3.2. `3161-ttc`

The `3161-ttc` COSE *protected* header parameter **MUST** be used for the mode described in Section 2.2.

The `3161-ttc` protected header parameter contains a DER-encoded TST [RFC3161] wrapped in a CBOR byte string (Major type 2).

The `MessageImprint` sent to the TSA (Section 2.4 of [RFC3161]) **MUST** be the hash of the payload of the COSE Signed Message. This does not include the `bstr` wrapping -- only the payload bytes. (For an example, see Appendix A.1.)

To minimize dependencies, the hash algorithm used for signing the COSE message **SHOULD** be the same as the algorithm used in the `MessageImprint` [RFC3161]. However, this may not be possible if the timestamp requester and the COSE message signer are different entities.

4. Timestamp Processing

Timestamp tokens [RFC3161] use Cryptographic Message Syntax (CMS) as the signature envelope format. [RFC5652] provides details about signature verification, and [RFC3161] provides details specific to timestamp token validation. The payload of the signed timestamp token is the `TSTInfo` structure defined in [RFC3161], which contains the `MessageImprint` that was sent to the TSA. The hash algorithm is contained in the `MessageImprint` structure, together with the hash itself.

As part of the signature verification, the receiver **MUST** make sure that the `MessageImprint` in the embedded timestamp token matches a hash of either the payload, signature, or signature fields, depending on the mode of use and type of COSE structure.

Appendix B of [RFC3161] provides an example that illustrates how timestamp tokens can be used to verify signatures of a timestamped message when utilizing X.509 certificates.

5. Security Considerations

Please review the Security Considerations section in [RFC3161]; these considerations apply to this document as well.

Also review the Security Considerations section in [RFC9052]. These considerations apply to this document as well, particularly with regard to the need for implementations to protect private key material. Additionally, solutions based on the COSE header parameters defined in this document must be able to report compromised keys promptly.

The following scenario assumes that an attacker can manipulate the clocks on the COSE signer and its relying parties, but not the TSA. It is also assumed that the TSA is a trusted third party, so the attacker cannot impersonate the TSA and create valid timestamp tokens. In such a setting, any tampering with the COSE signer's clock does not have an impact, because once the timestamp is obtained from the TSA, it becomes the only reliable source of time. However, in both CTT mode and TTC mode, a denial of service can occur if the attacker can adjust the relying party's clock so that the CMS validation fails. This could disrupt the timestamp validation.

In CTT mode, an attacker could manipulate the unprotected header by removing or replacing the timestamp. To avoid that, the COSE Signed Message should be integrity protected during transit and at rest.

In TTC mode, the TSA is given an opaque identifier (a cryptographic hash value) for the payload. While this means that the content of the payload is not directly revealed, to prevent comparison with known payloads or disclosure of identical payloads being used over time, the payload would need to be armored, e.g., with a nonce that is shared with the recipient of the header parameter but not the TSA. Such a mechanism is out of scope for this document.

The resolution, accuracy, and precision of the TSA clock, as well as the expected latency introduced by round trips to and from the TSA, must be taken into account when implementing solutions based on the COSE header parameters defined in this document.

5.1. Avoiding Semantic Confusion

CTT mode and TTC mode have different semantic meanings. An implementation must ensure that the contents of the CTT and TTC headers are interpreted according to their specific semantics. In particular, symmetric to the signature and assembly mechanics, each mode has its own separate verification algorithm.

Implementers **MUST** clearly differentiate between TSA timestamps [RFC3161] proving the existence of payload data at an earlier point in time (TTC) and timestamps explicitly providing evidence of the existence of the cryptographic signature (CTT). Failure to clearly distinguish between these timestamp semantics can result in vulnerabilities, such as incorrectly accepting signatures created after key revocation based on older payload-only timestamps. Validators must not interpret protected-header payload timestamps as proof of signature creation time and should rely exclusively on TSA timestamps [RFC3161] explicitly covering signature data for determining signature validity timing.

6. IANA Considerations

IANA has allocated the COSE header parameters defined in Table 1 in the "COSE Header Parameters" registry [IANA.cose_header-parameters] as follows:

Name	Label	Value Type	Value Registry	Description	Reference
`3161-ttc`	269	`bstr`	-	Timestamp Token per [RFC3161]: Timestamp, Then COSE	RFC 9921, Section 3.2
`3161-ctt`	270	`bstr`	-	Timestamp Token per [RFC3161]: COSE, Then Timestamp	RFC 9921, Section 3.1

Table 1: New COSE Header Parameters

7. Normative References

[IANA.cose_header-parameters] IANA, "COSE Header Parameters", <https://www.iana.org/assignments/cose>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, DOI 10.17487/RFC3161, August 2001, <https://www.rfc-editor.org/info/rfc3161>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/info/rfc9052>.

Appendix A. Examples

A.1. TTC

The payload

```
'This is the content.'
```

is hashed using SHA-256 to create the following `TimeStampReq` object

```
SEQUENCE {
  INTEGER 1
  SEQUENCE {
    SEQUENCE {
      OBJECT IDENTIFIER sha-256 (2 16 840 1 101 3 4 2 1)
      NULL
      }
    OCTET STRING
      09 E6 38 D4 AA 95 FD 72 71 86 62 03 59 53 03 BC
      E2 32 F4 62 A9 4D 38 E3 93 77 3C D3 AA E3 F6 B0
    }
  BOOLEAN TRUE
  }
```

which is sent to the TSA.

A `TimeStampResp` containing the following TST is returned:

```
SEQUENCE {
  OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
  [0] {
    SEQUENCE {
      INTEGER 3
      SET {
        SEQUENCE {
          OBJECT IDENTIFIER sha-512 (2 16 840 1 101 3 4 2 3)
          NULL
          }
        }
      SEQUENCE {
        OBJECT IDENTIFIER tSTInfo (1 2 840 113549 1 9 16 1 4)
        [0] {
          OCTET STRING, encapsulates {
            SEQUENCE {
              INTEGER 1
              OBJECT IDENTIFIER '1 2 3 4 1'
              SEQUENCE {
                SEQUENCE {
                  OBJECT IDENTIFIER sha-256 (2 16 840 1 101 3 4 2 1)
                  NULL
                  }
                OCTET STRING
                  09 E6 38 D4 AA 95 FD 72 71 86 62 03 59 53 03 BC
                  E2 32 F4 62 A9 4D 38 E3 93 77 3C D3 AA E3 F6 B0
                }
              INTEGER 12096870
              GeneralizedTime 29/08/2025 07:45:46 GMT
              BOOLEAN TRUE
[...]
```

The contents of the TST are `bstr`-wrapped and added to the protected headers bucket, which is then signed alongside the original payload to obtain the `COSE_Sign1` object.

```
18([
        <<{1: -7, 269: h'308215490609 2a864886f70d010702a082153a308215
36020103310f300d0609608648016503040203050030820184060b2a864886f70d010
9100104a08201730482016f3082016b02010106042a0304013031300d060960864801
650304020105000420 09e638d4aa95fd727186620359530 3bce232f462a94d38e3937
73cd3aae3f6b0020400b89566180f32303235303832393037343534365a0101ffa082
0111a482010d308201093111300f060355040a13084672656520545341310c300a060
355040b13035453413176307406035504 0d136d5468697320636572746966696361 74
6520646967697461 6c6c79207369676e7320646f63756d656e747320616e642074696
d65207374616d702072657175657374 73206d616465207573696e6720746865206672
65657473612e6f7267206f6e6c696e652073657276696365733118301606035504031
30f7777772e667265657473612e6f726731223020060 92a864886f70d010901161362
7573696c657a617340676d61696c2e636f6d311230100603550407130957756572 7a6
2757267310b3009060355040613024453 10f300d060355040813064261796572 6ea0
821008308208013082 05e9a003020102020900c1e986160da8e982300d06092a86488
6f70d01010d05003081953111300f060355040a13084672656520545341311 0300e06
0355040b1307526f6f7420434131183016060355040313 0f7777772e6672656574736
12e6f72673122302006092a864886f70d010901161362757 3696c657a617340676d61
696c2e636f6d311230100603550407130957756572 7a62757267310f300d060355040
8130642617965726e310b3009060355040613024445301e170d313630333133303135
3733395a170d323630333131313031353733395a308201093111300f060355040a13084
672656520545341310c300a060355040b13035453413176307406035504 0d136d5468
697320636572746966696361746520646967697461 6c6c79207369676e7320646f637
56d656e747320616e642074696d65207374616d702072657175657373206d616465
207573696e672074686520667265657473612e6f7267206f6e6c696e6520736572766
963657331183016060355040313 0f7777772e667265657473612e6f72673122302006
092a864886f70d010901161362757 3696c657a617340676d61696c2e636f6d3112301
006035504071309577565727a62757267310b3009060355040613024445310f300d06
035504081306426179657265 6e30820222300d06092a864886f70d01010105000382020
f003082020a0282020100b591048c4e486f34e9dc08627fc2375162236984b82cb130
beff517cfc38f84bce5c65a874dab2621ae0bce7e33563e0ede934fd5f8823159f078
48808227460c1ed88261706f4281334359dfbb81bd1353fc179610af1a8c8c865dc00
ea23b3a89be6bd03ba85a9ec827d60565905e22d6a584ed1380ae150280cee397e98a
012f380464007862443bc077cb95f421af31712d9683cdb6dffbaf3c8ba5ba566ae52
3d459d6177346d4d840e27886b7c01c5b890d78a2e27bba8dd2f9a2812e157d62f921
c65962548069dcdb7d06de181de0e9570d66f87220ce28b628ab55906f3ee0c210f70
51e8f4858af8b9a92d09e46af2d9cba5bfcfad168cdf604491a4b06603b114caf7031
f065e7eeefa53c575f3490c059d2e32ddc76ac4d4c4c710683b97fd1be591bc610551
86d88f9a0391b307b6f91ed954daa36f9acd6a1e14aa2e4adf17464b54db18dbb6ffe
300802465473704 36ce4e77bae5de6fe0f3f9d6e7ffbeb461e794e92fb0951f8aae61
a412cce9b21074635c8be327ae1a0f6b4a646eb0f8463bc63bf845530435d19e80251
1ec9f66c3496952d8becb69b0aa4d4c41f60515fe7dcbb89319cdda59ba6aea4be3ce
ae718e6fcb6ccd7db9fc50bb15b12f3665b0aa307289c2e6dd4b111ce48ba2d9efdb5
a6b9a506069334fb34f6fc7ae330f0b34208aac80df3266fdd90465876ba2cb898d95
05315b6e7b0203010001a38201db308201d730090603551d1304023000301d0603551
d0e041604146e760b7b4e4f9ce160ca6d2ce927a2a294b37737301f0603551d230418
30168014fa550d8c346651434cf7e7b3a76c95af7ae6a497300b0603551d0f0404030
206c030160603551d250101ff040c300a06082b06010505070308306306082b060105
0507010104573055302a06082b06010505073002861e687474703a2f2f7777772e667
265657473612e6f72672f7473612e6372743027060 82b06010505073001861b687474
703a2f2f7777772e667265657473612e6f72673a3235363030370603551d1f0430302
e302ca02aa0288626687474703a2f2f7777772e667265657473612e6f72672f63726c
2f726f6f745f63612e63726c3081c60603551d200481be3081bb3081b80601003081b
2303306082b06010505070201162768747 4703a2f2f7777772e667265657473612e6f
72672f667265657473615f6370732e68746d6c303206082b06010505070201162668 7
474703a2f2f7777772e667265657473612e6f72672f667265657473615f6370732e70
6466304706082b06010505070202303b1a3946726565545341207472757374656420 7
```

```
4696d657374616d70696e6720536f6674776172652061732061205365727669636520
28536161532930 0d06092a864886f70d01010d05000382020100a5c944e2c6fac0a14
d930a7fd0a0b172b41fc1483c3e957c68a2bcd9b9764f1a950161fd72472d41a5eed2
77786203b5422240fb3a26cde176087b6fb1011df4cc19e2571aa4a051109665e94c4
6f50bd2adee6ac4137e251b25a39dabda451515d8ff9e07209e8ec20b7874f7e1a0ed
e7c00937fe84a334f8b3265ced2d8ed9df61396583677feb382c1ee3b23e6ea5f05df
30de7b9f89005d25266f612f39c8b4f6daba6d7bfbac19632b90637329f52a6f066a1
0e43eaa81f849a6c5fe3fe8b5ea23275f687f2052e502ea6c30762a668cce07871dd8
e97e315bba929e25589977a0a312ce96c5106b1437c779f2b361b182888f3ee8a2343
74fa063e956192627f7c431073965d1260928eba009e803429ae324cf96f042354f37
bca5afddc79f79346ab388bfc79f01dc9861254ea6cc129941076b83d20556f3be513
26837f2876f7833b370e7c3d410523827d4f53400c72218d75229ff10c6f8893a9a3a
1c0c42bb4c898c13df41c7f6573b4fc56515971a610a7b0d2857c8225a9fb204eacec
a2e8971aa1af87886a2ae3c72fe0a0aae842980a77bef16b92115458090d982b59466
03764e75a0ad3d11454b9986f678b9ab6afe8497033ae3abfd4eb43b7bc9dee688159
49e6481582a82e785277f2282107efe390200e0508acb8ea82ea2505276f3c9da2a3d
3b4ad38bbf8842bda36fc2448291f558dc02dd1e0308207ff308205e7a00302010202
0900c1e986160da8e980300d06092a864886f70d01010d05003081953111300f06035
5040a13084672656520545341311030 0e060355040b1307526f6f7420434131183016
0603550403130f7777772e667265657473612e6f72673122302006092a864886f70d0
109011613627573696c657a617340676d61696c2e636f6d311230100603550407130 9
577565727a62757267310f300d06035504081306426179657265 6e310b3009060355040
613024445301e170d313630333133303135323133 5a170d343130333037303135323 1
335a3081953111300f060355040a13084672656520545341311030 0e060355040b130
7526f6f7420434131183016060355040313 0f7777772e667265657473612e6f726731
22302006092a864886f70d0109011613627573696c657a617340676d61696c2e636f6
d311230100603550407130957756572 7a62757267310f300d06035504081306426179
65726e310b3009060355040613024445308202223 00d06092a864886f70d010101050
00382020f003082020a0282020100b6028e0e3032f11110d964cda94b9d0278e1942a
e913aaa59907cda69793995bd9ac7e33bad9fe3704da1c01a98d21afe3f591a59d706
7705167998f5016722e0ab462b21f439171d2cfcc4593f3735af794a5ab311f6c010c
7898de33d75c4510ee76f4bd1d1498cf17d303f06a5dd9f796cc6ca9b657a56fe3ea4
fefbe7ce6b6a18d3e35a30cee5ff170d1cf39a333d3fda8964d22db685b29e561be89
0f0aa845873b2e84ab26ab839ffe8fade9d23bb31e61d273cc9b880649185fabecfa0
534600aba901b614e2e854582dea2226fc19cd7df52bed50d8777cd9988c053a3fc7d
c3287a068a4ff12b713cd9803666e955385456ff38f80298cf6b93856e9224774a66c
f1cdd11c2f8efd85203d7458b25664b13ed639cded4ff8113d6cc5353d2729473c3c3
07157c722aa5b5dd0bfb2d6c38b1b93749c881ec60026d08951b3824bd71bacbce473
aebd636f0b918b4a2c8ff4694f07457af2d6f1cf82554d1770fd79ff5d314dcd104cd
dcabc94138056dfcf017e7eb8572fd52f70144f188da05f5823f58dd06297e7387bed
2d772c13da8266601045fe412dd70986c0c987ba7344b9037387516d258e7885b51f8
968b7f2601213bc4cb4c85f8ff0b84af6a988337cdfb81868f7ecf31dca6716d7ec2d
d802c1672629e5c0052cb357dd29aafc43f615b3b1ff9d4e1ce08c71c73e1febb7dc5
6a33621329e9ed6c230203010001a382024e3082024a300c0603551d1304053003010
1ff300e0603551d0f0101ff0404030201c6301d0603551d0e04160414fa550d8c3466
51434cf7e7b3a76c95af7ae6a4973081ca0603551d230481c23081bf8014fa550d8c3
46651434cf7e7b3a76c95af7ae6a497a1819ba4819830819531113 00f060355040a13
0846726565205453413110300e060355040b1307526f6f74204341311830160603550
403130f7777772e667265657473612e6f72673122302006092a864886f70d01090116
13627573696c657a617340676d61696c2e636f6d311230100603550407130957756 57
27a62757267310f300d06035504081306426179657265 6e310b3009060355040 6130244
45820900c1e986160da8e98030330603551d1f042c302a3028a026a02486226874747
03a2f2f7777772e667265657473612e6f72672f726f6f745f63612e63726c3081cf06
03551d200481c73081c43081c1060a2b0601040181f22401013081b2303306082b060
1050507020116276874747 03a2f2f7777772e667265657473612e6f72672f66726565
7473615f6370732e68746d6c303206082b0601050507020116266874747 03a2f2f777
7772e667265657473612e6f72672f667265657473615f6370732e706466304706082b
0601050507020230 3b1a3946726565545341207472757374656420746 96d657374616
```

```
d70696e6720536f6674776172652061732061205365727669636520285361615329303706082b06010505070101042b3029302706082b06010505073001861b687474703a2f2f7777772e667265657473612e6f72673a32353630300d06092a864886f70d01010d050003820201068af7ebf938562ef4ceb3b580be2faf6cc35a26772962f3d95901fa5630c87d09198984ce8a06a33f8a9c282ed9f1cb11ac6c23e17108ee4efce6fb294de95c133262255725522ca61971d4a3b7f78250dfb8d4aeec0fb1959b164100520b9c10e64c62662e4ad4d0abae2298fc948fc4e99e8d9e6b8fdbe4404121ec7c1422eacb2c9d7328e07396e60b4f3bb803ad4a555c80fefb53f85e7764a0a9fb4afc399f4cd2f5fbf587105c6081cf3d05337b6bb7d1b010b749f4888c912f3696ba1b6902d77b7dfc046c04a0cc1ec4f8d185e2da55dfb7bc2a2036c6219246a4f99ddbb6f1f829398f3b803dc0ad90dcb59bef4c27c77404b99043b78271867991152c399f12cbfc4c625adc096355ae44e342100ec517a502e2f06f940b8d43599bbc1154f8ae761a0b0d555fb4a1391d4f3420af8dbf12f2d7ddb9d77dce1537804074af175e4f2d6d55b34b5d6f7dcbdd31730af56480d4c0cff143f9e83bc151866d0ba0f0bbdc47fe27864176bbd6c1ab85df325edf777889bc4471bf3fa73e56cc591e8b160cda7b0786a1ec04ac3b24fa2e28d5d19e5e48004d5e166a83c82ec6fd54fb385ebaf7133a85b52de46db5244e1c34ae8d36e712f9fce0d493d7d3edd586c6198e3ec3e6e96346f417ac9f221e0aff33a8f6a0b1ef4c023630b76adaa8d91433825ecc41c49a5b98b181c7da30e997ab954c73c2cd805afda993182038a308203860201013081a33081953111300f060355040a13084672656520545341311030 0e060355040b1307526f6f7420434131183016060355040313 0f77777 72e667265657473612e6f72673122302006092a864886f70d010901161362757369 6c657a617340676d61696c2e636f6d3112301006035504071309577565727a627572673 10f300d06035504081306426179657265 6e310b300906035504061302444502090 0c1e9 86160da8e982300d06096086480165030402030500a081b8301a06092a864886f70d0 10903310d060b2a864886f70d0109100104301c06092a864886f70d010905310f170d 323530383239303734353436 5a302b060b2a864886f70d010910020c311c301a30183 0160414916da3d860ecca82e34bc59d1793e7e968875f14304f06092a864886f70d01 0904314204408831bbe259ac6314847a26804e155a6d04c485b43aa55d2dd4c6dd7f5 943b5bd5a3dd24f05a17a2658ef65759ce4e0001f2b8ff99e38718044ab3784b3f174 b6300d06092a864886f70d0101010500048202000b1536e5491e8e941364fa7f2640d 30bfc8543dd8d472dad2db6df0c0483633d2dd4b9455f05d6e65a48ad9382e03741fc be1e0c8f7f607bae33979f9f4f71a07d852db0869518733fafe60779867781f584fc5 221a4fbdfd0736e976543ff5170b5520a65ea8b0b04f8f92b39808b4e3ed74d66d63b a0d1db353baa829db1d1905e4e833fb8f3824acff1a18a4735e5381b89c5e0df92d16 ec0a9a552298c52027e7bde806c153c1161d466d706455c0ae32d0cb108ca86209f57 edc3a7f4b36215170994d9ecb9e69d31bab52567b84a3a1568540469984d9b5b6bf63 4f9d022999cbd6519516d53065f919bee0f520b6b539e2f8fca66f2590c1ce032cb5b fdb170ad32125372e651ca4fa7a05ac72f7d5814ea324f99ad2c8110c06853fcf7d2a f1f28543b0f9ceba2a0f1536faabb07587ebe1d1dddd59fc804697928276613f8d146 f966812da7f25748cfcd298891acdfe041632b760677dfd53865d04d186ce7735d119 0aee0b2cddc0c55e6c48acfda749ec20af4dc0739430d10388bc83efed192c22917f2 f4a67474ac5f36e6608bb71631803fd5fb1a78d7973dd2a01c84dda46f9befccebfcb 300ab73628716b8151acf94e58af15de27c141c8d5ef4f82a51bbebc54cb2e1d4ac2f 0c05be7d3db16b9687f5a2fd28fb110f78f82a0ad0370a16cd9cbb59dc0814cba99e1 11e33482e45c9b4f948bff15eba70'}>>,
        {4: '11'},
        'This is the content.',
        h'f5f0f27964f178dcb2254b30fdfdc48abc4499beaea7cb80f4004f30403f13a44bcca24fc61c5d71d3823bac04b923011dc7d31de35df1aefcd5a8ec5fe0fe6e'
    ])
```

A.2. CTT

Starting with the following `COSE_Sign1` object,

```
18(
  [
    / protected h'a10126' / << {
        / alg / 1:-7 / ECDSA 256 /
      } >>,
    / unprotected / {
      / kid / 4:'11'
    },
    / payload / 'This is the content.',
    / signature / h'8eb33e4ca31d1c465ab05aac34cc6b23d58fef5c083106c4d
25a91aef0b0117e2af9a291aa32e14ab834dc56ed2a223444547e01f11d3b0916e5a4
c345cacb36'
  ]
)
```

the CBOR-encoded signature field is hashed using SHA-256 to create the following `TimeStampReq` object

```
SEQUENCE {
  INTEGER 1
  SEQUENCE {
    SEQUENCE {
      OBJECT IDENTIFIER sha-256 (2 16 840 1 101 3 4 2 1)
      NULL
      }
    OCTET STRING
      DD 94 71 EF E7 43 C4 05 13 35 DF 8F 6D 28 82 F3
      BA DC 38 77 00 F7 ED 3F 70 91 67 2A 3E EA F7 C8
    }
  BOOLEAN TRUE
  }
```

which is sent to the TSA.

A `TimeStampResp` containing the following TST is returned:

```
SEQUENCE {
  OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
  [0] {
    SEQUENCE {
      INTEGER 3
      SET {
        SEQUENCE {
          OBJECT IDENTIFIER sha-512 (2 16 840 1 101 3 4 2 3)
          NULL
          }
        }
      SEQUENCE {
        OBJECT IDENTIFIER tSTInfo (1 2 840 113549 1 9 16 1 4)
        [0] {
          OCTET STRING, encapsulates {
            SEQUENCE {
              INTEGER 1
              OBJECT IDENTIFIER '1 2 3 4 1'
              SEQUENCE {
                SEQUENCE {
                  OBJECT IDENTIFIER sha-256 (2 16 840 1 101 3 4 2 1)
                  NULL
                  }
                OCTET STRING
                  DD 94 71 EF E7 43 C4 05 13 35 DF 8F 6D 28 82 F3
                  BA DC 38 77 00 F7 ED 3F 70 91 67 2A 3E EA F7 C8
                }
              INTEGER 12100074
              GeneralizedTime 29/08/2025 07:53:00 GMT
              BOOLEAN TRUE
[...]
```

The contents of the TST are `bstr`-wrapped and added to the unprotected headers bucket in the original `COSE_Sign1` object to obtain the following:

```
18(
  [
    / protected h'a10126' / << {
       / alg / 1:-7 / ECDSA 256 /
      } >>,
    / unprotected / {
      / 3161-ctt / 270 : h'3082154906092a864886f70d010702a082153a3082
1536020103310f300d0609608648016503040203050030820184060b2a864886f70d0
109100104a08201730482016f3082016b02010106042a0304013031300d0609608648
0165030402010500042 0dd9471efe743c4051335df8f6d2882f3badc387700f7ed3f7
091672a3eeaf7c8020400b8a1ea180f32303235303832393037353330305a0101ffa0
820111a482010d308201093111300f060355040a13084672656520545341310c300a0
60355040b13035453413176307406035504 0d136d5468697320636572746966696361
7465206469676974616c6c79207369676e7320646f63756d656e747320616e6420746
96d65207374616d702072657175657374732 06d6164520757369 6e67207468652066
7265657473612e6f7267206f6e6c696e652073657276696365733118301606035504 0
3130f7777772e667265657473612e6f726731223020060 92a864886f70d0109011613
627573696c657a617340676d61696c2e636f6d311230100603550407130957756572 7
a62757267310b3009060355040613024445310f300d060355040813064261796 5726e
a0821008308208013082 05e9a003020102020900c1e986160da8e982300d06092a864
```

```
886f70d01010d05003081953111300f060355040a13084672656520545341311030 0e
060355040b1307526f6f74204341311830160603550403130f7777772e66726565747
3612e6f72673122302006092a864886f70d010901161362757369 6c657a617340676d
61696c2e636f6d311230100603550407130957756572 7a62757267310f300d0603550
40813064261796572 6e310b3009060355040613024445301e170d3136303331333031
353733395a170d323630333131303135373339 5a3082010931111300f060355040a130
846726565205453413 10c300a060355040b13035453413176307406035504 0d136d54
686973206365727469666963617465206469676974616c6c79207369676e7320646f6
3756d656e747320616e642074696d65207374616d702072657175657374737320 6d6164
65207573696e672074686520667265657473612e6f7267206f6e6c696e65207365727
669636573311830160603550403130f7777772e667265657473612e6f726731223020
06092a864886f70d010901161362757369 6c657a617340676d61696c2e636f6d31123
010060355040713095775657 27a62757267310b3009060355040613024445310f300d
060355040813064261796572 6e30820222300d06092a864886f70d010101050003820
20f003082020a0282020100b591048c4e486f34e9dc08627fc2375162236984b82cb1
30beff517cfc38f84bce5c65a874dab2621ae0bce7e33563e0ede934fd5f8823159f0
7848808227460c1ed88261706f4281334359dfbb81bd1353fc179610af1a8c8c865dc
00ea23b3a89be6bd03ba85a9ec827d60565905e22d6a584ed1380ae150280cee397e9
8a012f380464007862443bc077cb95f421af31712d9683cdb6dffbaf3c8ba5ba566ae
523d459d6177346d4d840e27886b7c01c5b890d78a2e27bba8dd2f9a2812e157d62f9
21c65962548069dcdb7d06de181de0e9570d66f87220ce28b628ab55906f3ee0c210f
7051e8f4858af8b9a92d09e46af2d9cba5bfcfad168cdf604491a4b06603b114caf70
31f065e7eeefa53c575f3490c059d2e32ddc76ac4d4c4c710683b97fd1be591bc6105
5186d88f9a0391b307b6f91ed954daa36f9acd6a1e14aa2e4adf17464b54db18dbb6f
fe30080246547370436ce4e77bae5de6fe0f3f9d6e7ffbeb461e794e92fb0951f8aae
61a412cce9b21074635c8be327ae1a0f6b4a646eb0f8463bc63bf845530435d19e802
511ec9f66c3496952d8becb69b0aa4d4c41f60515fe7dcbb89319cdda59ba6aea4be3
ceae718e6fcb6ccd7db9fc50bb15b12f3665b0aa307289c2e6dd4b111ce48ba2d9efd
b5a6b9a506069334fb34f6fc7ae330f0b34208aac80df3266fdd90465876ba2cb898d
9505315b6e7b0203010001a38201db308201d730090603551d1304023000301d06035
51d0e041604146e760b7b4e4f9ce160ca6d2ce927a2a294b37737301f0603551d2304
1830168014fa550d8c346651434cf7e7b3a76c95af7ae6a497300b0603551d0f04040
30206c030160603551d250101ff040c300a06082b06010505070308306306082b0601
0505070101045730553 02a06082b06010505073002861e687474703a2f2f7777772e6
67265657473612e6f72672f7473612e637274302706082b06010505073001861b6874
74703a2f2f7777772e667265657473612e6f72673a3235363030370603551d1f04303
02e302ca02aa028862687474703a2f2f7777772e667265657473612e6f72672f6372
6c2f726f6f745f63612e63726c3081c60603551d200481be3081bb3081b8060100308
1b2303306082b0601050507020116276874747 03a2f2f7777772e667265657473612e
6f72672f667265657473615f6370732e68746d6c303206082b06010505070201162 66
87474703a2f2f7777772e667265657473612e6f72672f667265657473615f6370732e
706466304706082b06010505070202303b1a394672656554534120747275737465642
074696d657374616d70696e6720536f667477617265206173206120536572766963 5
202853616153293 00d06092a864886f70d01010d05000382020100a5c944e2c6fac0a
14d930a7fd0a0b172b41fc1483c3e957c68a2bcd9b9764f1a950161fd72472d41a5ee
d277786203b5422240fb3a26cde176087b6fb1011df4cc19e2571aa4a051109665e94
c46f50bd2adee6ac4137e251b25a39dabda451515d8ff9e07209e8ec20b7874f7e1a0
ede7c00937fe84a334f8b3265ced2d8ed9df61396583677feb382c1ee3b23e6ea5f05
df30de7b9f89005d25266f612f39c8b4f6daba6d7bfbac19632b90637329f52a6f066
a10e43eaa81f849a6c5fe3fe8b5ea23275f687f2052e502ea6c30762a668cce07871d
d8e97e315bba929e25589977a0a312ce96c5106b1437c779f2b361b182888f3ee8a23
4374fa063e956192627f7c431073965d1260928eba009e803429ae324cf96f042354f
37bca5afddc79f79346ab388bfc79f01dc9861254ea6cc129941076b83d20556f3be5
1326837f2876f7833b370e7c3d410523827d4f53400c72218d75229ff10c6f8893a9a
3a1c0c42bb4c898c13df41c7f6573b4fc56515971a610a7b0d2857c8225a9fb204eac
eca2e8971aa1af87886a2ae3c72fe0a0aae842980a77bef16b92115458090d982b594
6603764e75a0ad3d11454b9986f678b9ab6afe8497033ae3abfd4eb43b7bc9dee6881
5949e6481582a82e785277f2282107efe390200e0508acb8ea82ea2505276f3c9da2a
```

```
3d3b4ad38bbf8842bda36fc2448291f558dc02dd1e0308207ff308205e7a003020102
020900c1e986160da8e980300d06092a864886f70d01010d05003081953111300f060
355040a13084672656520545341311030 0e060355040b1307526f6f74204341311830
160603550403130f7777772e667265657473612e6f726731223020060092a864886f70
d0109011613627573696c657a617340676d61696c2e636f6d311230100603550407130
09577565727a62757267310f300d060355040813064261796572 6e310b30090603550
40613024445301e170d313630333133303135323133 5a170d343130333037303135323
31335a3081953111300f060355040a13084672656520545341311030 0e060355040b1
307526f6f74204341311830160603550403130f7777772e667265657473612e6f7267
3122302006092a864886f70d0109011613627573696c657a617340676d61696c2e636
f6d311230100603550407130957756572 7a62757267310f300d060355040813064261
7965726e310b30090603550406130244530820222300d06092a864886f70d0101010
5000382020f003082020a0282020100b6028e0e3032f11110d964cda94b9d0278e194
2ae913aaa59907cda69793995bd9ac7e33bad9fe3704da1c01a98d21afe3f591a59d7
067705167998f5016722e0ab462b21f439171d2cfcc4593f3735af794a5ab311f6c01
0c7898de33d75c4510ee76f4bd1d1498cf17d303f06a5dd9f796cc6ca9b657a56fe3e
a4fefbe7ce6b6a18d3e35a30cee5ff170d1cf39a333d3fda8964d22db685b29e561be
890f0aa845873b2e84ab26ab839ffe8fade9d23bb31e61d273cc9b880649185fabecf
a0534600aba901b614e2e854582dea2226fc19cd7df52bed50d8777cd9988c053a3fc
7dc3287a068a4ff12b713cd9803666e955385456ff38f80298cf6b93856e9224774a6
6cf1cdd11c2f8efd85203d7458b25664b13ed639cded4ff8113d6cc5353d2729473c3
c307157c722aa5b5dd0bfb2d6c38b1b93749c881ec60026d08951b3824bd71bacbce4
73aebd636f0b918b4a2c8ff4694f07457af2d6f1cf82554d1770fd79ff5d314dcd104
cddcabc94138056dfcf017e7eb8572fd52f70144f188da05f5823f58dd06297e7387b
ed2d772c13da8266601045fe412dd70986c0c987ba7344b9037387516d258e7885b51
f8968b7f2601213bc4cb4c85f8ff0b84af6a988337cdfb81868f7ecf31dca6716d7ec
2dd802c1672629e5c0052cb357dd29aafc43f615b3b1ff9d4e1ce08c71c73e1febb7d
c56a33621329e9ed6c230203010001a382024e3082024a300c0603551d13040530030
101ff300e0603551d0f0101ff0404030201c6301d0603551d0e04160414fa550d8c34
6651434cf7e7b3a76c95af7ae6a4973081ca0603551d230481c23081bf8014fa550d8
c346651434cf7e7b3a76c95af7ae6a497a1819ba4819830819531113 00f060355040a
13084672656520545341311030 0e060355040b1307526f6f74204341311830160603 5
50403130f7777772e667265657473612e6f726731223020060092a864886f70d010901
1613627573696c657a617340676d61696c2e636f6d311230100603550407130957756
5727a62757267310f300d060355040813064261796572 6e310b300906035504061302
4445820900c1e986160da8e98030330603551d1f042c302a3028a026a024862268747
4703a2f2f7777772e667265657473612e6f72672f726f6f745f63612e63726c3081cf
0603551d200481c73081c43081c1060a2b0601040181f22401013081b2303306082b0
601050507020116276874747 03a2f2f7777772e667265657473612e6f72672f667265
657473615f6370732e68746d6c303206082b0601050507020116266874747 03a2f2f7
777772e667265657473612e6f72672f667265657473615f6370732e70646630470608
2b0601050507020230 3b1a3946726565545341207472757374656420746 96d6573746
16d70696e6720536f667477617265206173206120536572766963652028536161532 9
303706082b0601050507010104 2b3029302706082b0601050507300186 1b687474703
a2f2f7777772e667265657473612e6f72673a32353630300d06092a864886f70d0101
0d0500038202010068af7ebf938562ef4ceb3b580be2faf6cc35a26772962f3d95901
fa5630c87d09198984ce8a06a33f8a9c282ed9f1cb11ac6c23e17108ee4efce6fb294
de95c133262255725522ca61971d4a3b7f78250dfb8d4aeec0fb1959b164100520b9c
10e64c62662e4ad4d0abae2298fc948fc4e99e8d9e6b8fdbe4404121ec7c1422eacb2
c9d7328e07396e60b4f3bb803ad4a555c80fefb53f85e7764a0a9fb4afc399f4cd2f5
fbf587105c6081cf3d05337b6bb7d1b010b749f4888c912f3696ba1b6902d77b7dfc0
46c04a0cc1ec4f8d185e2da55dfb7bc2a2036c6219246a4f99ddbb6f1f829398f3b80
3dc0ad90dcb59bef4c27c77404b99043b78271867991152c399f12cbfc4c625adc096
355ae44e342100ec517a502e2f06f940b8d43599bbc1154f8ae761a0b0d555fb4a139
1d4f3420af8dbf12f2d7ddb9d77dce1537804074af175e4f2d6d55b34b5d6f7dcbdd3
1730af56480d4c0cff143f9e83bc151866d0ba0f0bbdc47fe27864176bbd6c1ab85df
325edf777889bc4471bf3fa73e56cc591e8b160cda7b0786a1ec04ac3b24fa2e28d5d
19e5e48004d5e166a83c82ec6fd54fb385ebaf7133a85b52de46db5244e1c34ae8d36
```

```
e712f9fce0d493d7d3edd586c6198e3ec3e6e96346f417ac9f221e0aff33a8f6a0b1e
f4c023630b76adaa8d91433825ecc41c49a5b98b181c7da30e997ab954c73c2cd805a
fda993182038a3082038602010130 81a33081953111300f060355040a130846726565
20545341311030 0e060355040b1307526f6f7420434131183016060355040313 0f777
7772e667265657473612e6f72673122302006092a864886f70d010901161362757369
6c657a617340676d61696c2e636f6d3112301006035504071309577565727a6275726
7310f300d060355040813064261796572 6e310b3009060355040613024445020900c1
e986160da8e982300d06096086480165030402030500a081b8301a06092a864886f70
d010903310d060b2a864886f70d0109100104301c06092a864886f70d010905310f17
0d323530383239303735333030 5a302b060b2a864886f70d010910020c311c301a301
830160414916da3d860ecca82e34bc59d1793e7e968875f14304f06092a864886f70d
010904314204401d3b1f355cc995b2c7a38dfee19a0815ae93a9078cea6db540501ee
df305e9f9f41349096a089bf5358380d6ed01eb508cbb551d120e9aca924429148ef1
a229300d06092a864886f70d0101010500048202004f22fe5e554c950f7f74462adde
4f7c4c412d60479c6950c2509d1a5063e04c284eb42dda42e3591447b63fdc72c953e
f04c81c1e59874c4d02cfb6b63de977d439998995e960a25755304a12ed23e7ccae97
678a3dd94bc4025399806c9d00454a740800d3dc13016143af48b80c1d24033694f2b
edb7c25d35c065e9c2fe71cee598ac2e8700bed5b755f001da3227f85fc178f27c565
64ef5ff64b874916ab6fd2d966c542936a9940d0a5685463dc8e5b6ee82d639abb683
433603541db3362ad77667e2ded4160c8f87e5c048d6bd05a7831871bb1052ddac132
f35baadc2ceea41834efd276d4d2a8525879bd909b3d930d3cd4ef1d87d1a5f47bd9b
ef00956fee8e55d2d40b7447074a7295b204f07ee086775729d9cdb59407956127223
88cb3af8a96fac65c79179c7e5292ce06e3f582e3f7d8fa6d7d41759bbd593b32a0fa
c8149a2b015e795fca2810133c2d768ef8d9da66ba192cbf142d2e4571e491ed7f7b0
eb920f22c4492ba0260d30fef98a4d503693afe3dcc561b04bb3b32d8a49f27f988fe
faa5f7b1af110bdad64a2825348a46651e1371e625c9792dfe9780528e5eb17f6078f
cb418a420129e7a19bf8f27508b256e755753d8e6b436c384fa350c2e4e9018fd372c
f54f303d462832675c8ac89f04c360a1d0d82f8d52ff7d815e74ad4aa19a68a9acfd2
450855dcb3b2a528063d426dc30268f',
      / kid / 4:'11'
    },
    / payload / 'This is the content.',
    / signature / h'8eb33e4ca31d1c465ab05aac34cc6b23d58fef5c083106c4
d25a91aef0b0117e2af9a291aa32e14ab834dc56ed2a223444547e01f11d3b0916e5
a4c345cacb36'
  ]
)
```

Acknowledgments

The authors would like to thank Alexey Melnikov, Carl Wallace, Carsten Bormann, Deb Cooley, Éric Vyncke, Francesca Palombini, Leonard Rosenthol, Linda Dunbar, Michael B. Jones, Michael Prorock, Mike Bishop, Mohamed Boucadair, Orie Steele, Roman Danyliw, Shuping Peng, Stefan Santesson, Steve Lasker, and Yingzhen Qu for their reviews and comments.

Contributors

Carsten Bormann
Email: cabo@tzi.org

Carsten contributed part of the security considerations.

Orie Steele
Email: orie@transmute.industries

Orie contributed an improved version of the diagrams.

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
64295 Darmstadt
Germany
Email: henk.birkholz@ietf.contact

Thomas Fossati
Linaro
Email: thomas.fossati@linaro.org

Maik Riechert
Microsoft
United Kingdom
Email: Maik.Riechert@microsoft.com