19. RFC 8152：CBOR Object Signing and Encryption (COSE)
RFC 8152：CBOR 物件簽署和加密（COSE）

網際網路工程任務組（IETF）
Request for Comments：8152
分類：標準
ISSN：2070-1721

J. Schaad
August Cellars
2017 年 7 月

CBOR 物件簽署和加密（COSE）

摘要

簡明二進制物件表示法（CBOR）是一種專為小程式碼和小訊息而設計的資料格式。需要能夠為此資料格式定義基本安全服務。本文定義 CBOR 物件簽署和加密（COSE）協定。本規範描述如何使用 CBOR 進行序列化來建立和處理簽章，訊息鑑別碼和加密。該規範也描述如何使用 CBOR 表示加密密鑰。

本文的狀態

這是網際網路標準跟蹤文件。

本文是網際網路工程任務組（IETF）的產品。它代表 IETF 社群的共識。它已經過公眾審查，並已獲得網際網路工程指導組（IESG）的批准發布。有關網際網路標準的更多信息，請參見 RFC 7841 的第 2 節。

有關本文當前狀態，任何勘誤以及如何提供反饋的信息，請訪問 http://www.rfc-editor.org/info/rfc8152。

版權聲明

目錄

1. 前言

人們越來越關注構成物聯網（IoT）的小型受限制設備。該過程產生的標準之一是“簡明二進制物件表示法(CBOR)”[RFC7049]。CBOR 允許二進制資料以及其他更改來擴充 JavaScript 物件表示法（JSON）[RFC7159]的資料模型。一些處理物聯網的 IETF 工作小組正採用 CBOR 作為資料結構的編碼。CBOR 專門為在訊息傳輸和實現的方面都較小，並且是無網目解碼器而設計。需要為 IoT 提供訊息安全服務，並且使用 CBOR 作為訊息編碼格式是有意義的。

JOSE 工作小組使用 JSON 生成一組文件[RFC7515] [RFC7516] [RFC7517] [RFC7518]，它指定如何處理加密，簽章和訊息鑑別碼（MAC）操作以及如何使用 JSON 編碼鍵值。本文定義 CBOR 物件簽章和加密（COSE）標準，該標準對 CBOR 編碼格式執行相同的操作。雖然強烈嘗試保留原始 JSON 物件簽章和加密(JOSE)文件的風格，但需要考慮兩個因素：

- CBOR 具有 JSON 中不存在且適合使用的功能。這方面的例子是 CBOR 有一種直接編碼二進制的方法，而不需先將其轉換為 base64 編碼的字串。

- COSE 不是 JOSE 規範的直接副本。在創建 COSE 的過程中，將重新審查為 JOSE 做出的決策。在許多情況下，由於標準並不總是相同，因此決定了不同的結果。

1.1. JOSE 的設計變更

- 定義單一頂端的訊息結構，以便可以輕鬆辨識加密，簽章和 MAC 訊息，並且仍然具有一致的視圖。
- 簽章訊息將與內容相關受保護和不受保護的參數跟與簽名相關的參數區分開。
- MACed 訊息與簽章訊息分開。
- MACed 訊息能夠使用與封裝訊息同一組接收者演算法來獲取 MAC 認證金鑰。
- 對二進制資料使用二進制編碼而不是使用 base64url 編碼。
- 將加密演算法的驗證標籤與密文結合使用。
- 這組加密演算法已在某些方向上進行擴充，並在其他方向上進行修整。

1.2. 需求術語

關鍵字“必須(MUST)”，“不得(MUST NOT)”，“必要(REQUIRED)”，“必須(SHALL)”，“不應該(SHALL NOT)”，“應該(SHOULD)”，“不應該(SHOULD NOT)”，“建議(RECOMMENDED)”，“不建議(NOT RECOMMENDED)”，“可以(MAY)”，“可選(OPTIONAL)”在本文中，只有當它們皆為大寫字母時才會按照 BCP 14 [RFC2119] [RFC8174]中的描述進行解釋，如此處所示。

當單詞以小寫字母顯示時，此解釋不適用。

1.3. CBOR 語法

目前沒有可供規範使用的標準 CBOR 語法。因此，CBOR 結構用散文描述。

該文的發展首先是研究語法，然後開發散文。這樣的散文是使用由 CBOR 資料定義語言（CDDL）[CDDL]定義的原始型態字串編寫的。在本規範中，使用以下原始型別：

any -- 允許將所有 CBOR 值放在此處的非特定值。
bool -- 布林值(true：主要型態 7，值 21；false：主要型態 7，值 20）。
bstr -– 位元組字串（主要型態 2）。
int -- 無符號整數或負整數。
nil -- 空值（主要型態 7，值 22）。
nint -- 負整數（主要型態 1）。
tstr -- UTF-8 文本字串（主要型態 3）。
uint -- 無符號整數（主要型態 0）。

CDDL 中的兩種語法在本文中以速記形式出現。這些是：

FOO / BAR -- 表示此處可以顯示 FOO 或 BAR。
[+ FOO] -- 表示型態 FOO 在陣列中出現一次或多次。

除了散文描述之外，還有一個 CBOR 語法的版本以 CDDL 形式呈現。由於 CDDL 尚未在 RFC 中發布，因此該語法可能無法與 CDDL 的最終版本一起使用。 CDDL 語法是資訊性的；散文描述是規範性的。

可以通過下面的 XPath 表達式從本文的 XML 版本中提取收集的 CDDL。（根據正在使用的 XPath 評估程序，可能需要處理＆gt；作為實體。）

//artwork[@type='CDDL']/text()

CDDL 期望初始的非終端符號是文件中的第一個符號。出於這個原因，這裡介紹 CDDL 的第一個片段。

```
start = COSE_Messages / COSE_Key / COSE_KeySet / Internal_Types
```

這使工具被定義的更安定：

```
Internal_Types = Sig_structure / Enc_structure / MAC_structure /
        COSE_KDF_Context
```

定義非終端內部型態用於處理在編寫本文期間使用的自動驗證工具。它引用那些用於安全計算但不用於傳輸發送的非終端。

1.4. 與 CBOR 相關的術語

在 JSON 中，映射被稱為物件，映射鍵值只有一種：字串。在 COSE 中，我們使用字串，負整數和無符號整數作為映射鍵值。整數用於編碼的緊湊性和易於比較。包含字串允許使用額外範圍的短編碼值。由於“金鑰”一詞主要用於其他含義，因此我們使用術語“標籤”作為映射鍵值。

COSE 映射中呈現不是字串或整數的標籤是錯誤的。應用程式可能無法處理或處理帶有錯誤標籤的消息。；但是，他們不能建立帶有不正確的訊息的標籤。
CDDL 語法片段定義非終端“標籤”，如前一段所述，以及“值”，它允許使用任何值。

```
label = int / tstr
values = any
```

1.5. 文件術語

在本文中，我們使用以下術語：

Byte 是 octet 的同義字為八位元組。

受限制的應用協定（CoAP）是一種用於受約束系統的專用網傳輸協定。它在[RFC7252]中定義。

認證加密（AE）[RFC5116]演算法是那些使用加密服務提供內容演算法的認證檢查的加密演算法。

經過身份驗證的資料進行認證加密（AEAD）[RFC5116]演算法提供與 AE 演算法相同的內容身份驗證服務，但它們還提供對非加密資料的身份驗證。

2. 基本 COSE 結構

COSE 物件結構的設計使得在解析和處理不同型態的安全訊息時可能存在大量常用編碼。所有訊息結構都基於 CBOR 陣列型態構建。陣列的前三個元素始終包含相同的資訊：

1. 包含在 bstr 中一組受保護標頭參數。
2. 一組未受保護的標頭參數集合作為映射。
3. 訊息的內容。內容可以是明文或密文。內容可能已分離，但仍使用該位置。當存在時，內容被包裝在 bstr 中，並且在分離時是零值。

此後的元素取決於特定的訊息型態。
COSE 訊息也使用層的概念來構建以分離不同型態的加密概念。作為其工作原理的範例，請考慮 COSE_加密訊息（第 5.1 節）。此訊息類型分為兩層：內容層和接收層。在內容層中，明文被加密並且資訊有關放置加密訊息。在接收者層中，內容加密金鑰（CEK）被加密，並且資訊有關如何為每個接收者放置加密的信息。對於 CEK 預先共享的情況，提供加密訊息 COSE_Encrypt0（第 5.2 節）的單層版本。

通過以下方法確定已呈現的訊息類型：
1. 從上下文中可以知道特定的訊息類型。這可以通過包含結構中的標籤或由應用協定指定的限制來定義。
2. 訊息類型由 CBOR 標籤辨識。具有 CBOR 標籤的訊息在本規範中稱為標記訊息，而沒有 CBOR 標籤的訊息稱為未標記訊息。本文為每個訊息結構定義一個 CBOR 標籤。這些標籤可以在表 1 中找到。
3. 當 COSE 物件在多媒體型態“application / cose”中攜帶時，可選參數“cose-type”可用於辨識嵌式物件。如果使用結構的標記版本，則參數為 OPTIONAL。如果使用結構的未標記

版本，則該參數為 REQUIRED。表 1 中列出每個結構的參數值。

4. 當 COSE 物件作為 CoAP 負載攜帶時，CoAP 內容格式選項可用於辨別訊息內容。 CoAP 內容格式值可以在表 26 中找到。訊息結構的 CBOR 標籤不是必需的，因為每個安全訊息都是唯一辨識。

CBOR Tag	cose-type	Data Item	Semantics
98	cose-sign	COSE_Sign	COSE Signed Data Object
18	cose-sign1	COSE_Sign1	COSE Single Signer Data Object
96	cose-encrypt	COSE_Encrypt	COSE Encrypted Data Object
16	cose-encrypt0	COSE_Encrypt0	COSE Single Recipient Encrypted Data Object
97	cose-mac	COSE_Mac	COSE MACed Data Object
17	cose-mac0	COSE_Mac0	COSE Mac w/o Recipients Object

表 1：COSE 訊息辨別

以下 CDDL 片段辨識本文中定義的所有頂端訊息。為標記和未標記的訊息版本定義單獨的非終端。

```
COSE_Messages = COSE_Untagged_Message / COSE_Tagged_Message

COSE_Untagged_Message = COSE_Sign / COSE_Sign1 /
    COSE_Encrypt / COSE_Encrypt0 /
    COSE_Mac / COSE_Mac0

COSE_Tagged_Message = COSE_Sign_Tagged / COSE_Sign1_Tagged /
    COSE_Encrypt_Tagged / COSE_Encrypt0_Tagged /
    COSE_Mac_Tagged / COSE_Mac0_Tagged
```

3. 標頭參數

COSE 的結構被設計為具有兩個不被認為是負載本身的一部分的儲存區，但是用於保存關於使用在處理層的內容，演算法，密鑰或評估提示的資訊。除密鑰外，這兩個儲存區可用於所有結構。雖然存在這些儲存區，但它們可能並非在所有情況下都可用。例如，雖然受保護的儲存區被定義為接收者結構的一部分，但用於

接收者結構的某些演算法不提供經過驗證的資料。如果是這種情況，則受保護的儲存區將保留為空。

兩個儲存區都實現為CBOR映射。映射鍵值是‘標籤’（第1.4節）。值的部分取決於標籤的定義。兩個映射都使用相同的標籤/值對集合。標籤的整數和字串值已分為幾個部分，包括標準範圍，私有範圍和取決於所選演算法的範圍。已定義的標籤可在“COSE標頭參數”IANA註冊表（第16.2節）中找到。

每層提供兩個儲存區：

保護（protected）：包含有關當前層要加密保護的參數。如果它不會包含在加密計算中，則該儲存區必須為空。此儲存區在訊息中編碼為二進制物件。通過 CBOR 對受保護映射進行編碼並將其包裝在 bstr 物件中來獲取此值。發送者應該將零長度映射編碼為零長度字串，而不是零長度映射（編碼為 h'a0'）。零長度二進制編碼是優先的，因為它更短並且在序列化結構中用於加密計算的版本。對映射進行編碼後，該值將包裝在二進制物件中。接收者必須接受零長度二進制值和以二進制值編碼的零長度映射。包裝允許傳輸受保護映射的編碼，更有可能在傳輸過程中不會改變。（表現不佳的中間體可以解碼和重新編碼，但這將導致無法驗證，除非重新編碼的位元組字串與解碼的位元組字串相同。）這避免了所有各方需要能夠進行常見的規範編碼的問題。

未保護（unprotected）：包含有關當前層的未受加密保護的參數。

只有處理當前層的參數才會放在該層上。作為範例，參數“內容型態”描述訊息中攜帶的訊息的內容。因此，此參數僅放置在內容層中，而不是放在接收者或簽章層中。原則上，應該能夠處理任何給定的層而不參考任何其他層。除 COSE_標記結構外，唯一需要跨層的資料是加密密鑰。

儲存區存在於本文中定義的所有安全物件中。按順序排列的欄位是“protected”儲存區（作為 CBOR“bstr”型態），然後是“unprotected”儲存區（作為 CBOR“映射”型態）。需要存在兩個儲存區。進入儲存區的參數來自 IANA“COSE 標頭參數”註冊表（第 16.2 節）。一些常見參數在下一節中定義，但在本文中定義許多參數。

每個映射中的標籤必須是唯一的。處理訊息時，如果標籤出現多次，則必須拒絕訊息格式錯誤。應用程式應該驗證在受保護和不受保護的標頭中都不會出現相同的標籤。如果訊息未被格式化，則必須從受保護的儲存區中獲取屬性，然後才能從未受保護的儲存區中獲取屬性。

以下 CDDL 片段表示兩個標頭儲存區。在 CDDL 中定義一組“標頭”，表示放置屬性的兩個儲存區。該組用於在所有位置始終如一地提供這兩個欄位。還定義一種型態，表示常見標頭的映射。

```
Headers = (
   protected : empty_or_serialized_map,
   unprotected : header_map
)

header_map = {
   Generic_Headers,
   * label => values
}

empty_or_serialized_map = bstr .cbor header_map / bstr .size 0
```

3.1. 常見的 COSE 標頭參數

本節定義一組常見標頭參數。這些參數的摘要可以在表 2 中找到。應該參考該表來確定標籤的值和值的型態。

本節中定義的標頭參數集合為：

alg：此參數用於指示用於安全性處理的演算法。必須在存在此功能的情況下對此參數進行身份驗證。此支援由 AEAD 演算法或結構（COSE_Sign，COSE_Sign0，COSE_Mac 和 COSE_Mac0）提供。可以通過將標頭放在受保護的標頭儲存區中或作為外部提供的資料的一部分來完成此驗證。該值取自“COSE 算法”註冊表（參見第 16.4 節）。

crit：該參數用於指示處理訊息的應用程式需要了解哪些受保護的標頭標籤。不需要包含本文中定義的參數，因為所有實現都應該理解它們。如果存在，此參數必須放在受保護的標頭儲存區中。陣列必須至少有一個值。並非所有標籤都必須包含

在‘crit’參數中。決定哪些標題標籤放置在陣列中的規則為：

* 應該省略 0 到 8 範圍內的整數標籤。
* 可以省略 -1 到 -128 範圍內的整數標籤，因為它們與演算法有關。如果應用程式可以正確處理演算法，則可以假設它將正確處理與該演算法相關的所有公共參數。應該包括 -129 到 -65536 範圍內的整數標籤，因為這些參數可能為不會得到普遍支援的少見參數。
* 可省略申請所需參數的標籤。如果可以省略標籤，應用程式應該有一個聲明。

由‘crit’指示的標頭參數值可以由安全的程式庫或使用安全的應用程式庫處理；唯一的要求是處理參數。如果‘crit’值列表所包含參數不在受保護儲存區中的值，則這是處理訊息時的致命錯誤。

內容類型（content type）：此參數用於指示負載或密文欄位中資料的內容類型。整數來自“CoAP 內容格式”IANA 註冊表[COAP.Formats]。文本值遵循“<type-name> / <subtype-name>”的語法，其中<type-name>和<subtype-name>在[RFC6838]的第 4.2 節中定義。前導和尾隨空格也被省略。文本內容值隨著參數和子參數可以使用IANA“媒體類型”註冊表找到。如果內容結構可能不明確，應該提供此參數。

kid：此參數辨識可用作輸入的一段資料，以查找所需的加密密鑰。可以將此參數的值與 COSE_Key 結構中的‘kid’成員進行匹配。其他密鑰分配方法可以定義要匹配的等值欄位。應用程式絕不能假設‘kid’值是唯一的。可能存在多個具有相同‘kid’值的鍵值，因此可能需要檢查與該‘kid’相關聯的所有鍵值。‘kid’值的內部結構未定義，應用程式無法依賴。密鑰辨識符值是關於使用哪個密鑰的提示。這不是一個安全關鍵領域。因此，它可以放在未受保護的標頭儲存區中。

IV：該參數保存初始向量（IV）值。對於一些對稱式加密演算法，這可以稱為隨機數。 IV 可以放置在未受保護的標頭中，因為修改 IV 將導致解密產生易於檢測為亂碼的明文。

部分 IV（Partial IV）：該參數保存 IV 值的一部分。使用 COSE_Encrypt0 結構時，IV 的一部分可以是與密鑰關聯的上下文的一部分。該欄位用於攜帶的值時導致 IV 更改每條訊息。 IV 可以放置在未受保護的標頭中，因為修改 IV 將導致解密產生易於檢測為亂碼的明文。“初始化向量”和“部分初始化向量”參數不得同時存在於同一安全層中。

訊息 IV 由以下步驟生成：

1. 用零左填充部分 IV 到 IV 的長度。
2. 對填充的部分 IV 與上下文 IV 進行互斥或。

計數器簽章（counter signature）：此參數包含一個或多個計數器簽章值。計數器簽章提供使第二方簽署某些資料的方法。計數器簽章參數可以作為以下任何結構中的不受保護的屬性出現：COSE_Sign1，COSE_Signature，COSE_Encrypt，COSE_recipient，COSE_Encrypt0，COSE_Mac 和 COSE_Mac0。這些結構都具有相同的起始元素，因此可以計算計數器簽章的一致計算。有關計算計數器簽章的細節，請參見第 4.5 節。

Name	Label	Value Type	Value Registry	Description
alg	1	int / tstr	COSE Algorithms registry	Cryptographic algorithm to use
crit	2	[+ label]	COSE Header Parameters registry	Critical headers to be understood
content type	3	tstr / uint	CoAP Content-Formats or Media Types registries	Content type of the payload
kid	4	bstr		Key identifier
IV	5	bstr		Full Initialization Vector
Partial IV	6	bstr		Partial Initialization Vector
counter signature	7	COSE_Signature / [+ COSE_Signature]		CBOR-encoded signature structure

表 2：常見的標頭參數

下面給出表示本節中定義的標頭集合的 CDDL 片段。每個標題都標籤為可選，因為它們不需要位於每個映射中；上面討論特定映射中所需的標題。

```
Generic_Headers = (
   ? 1 => int / tstr,    ; algorithm identifier
   ? 2 => [+label],      ; criticality
   ? 3 => tstr / int,    ; content type
   ? 4 => bstr,          ; key identifier
   ? 5 => bstr,          ; IV
   ? 6 => bstr,          ; Partial IV
   ? 7 => COSE_Signature / [+COSE_Signature] ;
   Counter signature
)
```

4. 簽署物件

COSE 支援兩種不同的簽章結構。COSE_Sign 允許將一個或多個簽章應用於相同的內容。 COSE_Sign1 僅限於一個簽章者。結構

不能相互轉換；由於簽章計算包括結構正在使用的是哪個參數辨識，因此轉換後的結構將失敗簽章驗證。

4.1. 與一個或多個簽章者簽約

COSE_Sign 結構允許將一個或多個簽章應用於訊息負載。與簽章有關的內容和與簽章有關的參數與簽章本身一起被攜帶。這些參數可以通過簽章進行驗證，也可以僅存在。關於內容的參數的範例是內容類型。關於簽章的參數的範例將是用於建立簽章和計數器簽章的演算法和密鑰。

RFC 5652 表明：

> 當存在多個簽章時，與特定簽章者相關聯的一個成功驗證的簽章通常被該簽章者視為成功簽章。但是，有些應用程式環境需要其他規則。對每個簽章者使用一個有效簽章之外的規則的應用程式必須指定這些規則。此外，在簽章者識別符的簡單匹配不足以確定簽章是否由同一簽章者生成的情況下，應用程式規範必須描述如何確定哪些簽章是由同一簽章者生成的。支援不同的接收者社群是簽章者選擇包含多個簽章的主要原因。

例如，COSE_Sign 結構可能包括使用 Edwards 曲線數位簽章演算法（EdDSA）[RFC8032]和橢圓曲線數位簽章演算法（ECDSA）[DSS]生成的簽章。這允許接收者驗證與一種演算法或另一種演算法相關聯的簽章。有關多個簽章評估的更詳細資訊可以在[RFC5752]中找到。

簽章結構可以編碼為標籤或未標籤，具體取決於它將使用的上下文。標籤的 COSE_Sign 結構由 CBOR 標籤 98 辨別。表示此的 CDDL 片段是：
COSE 簽章訊息分為兩部分。攜帶正文和有關正文訊息的 CBOR 物件的稱為 COSE_Sign 結構。攜帶簽章和有關簽章的訊息的 CBOR 物件稱為 COSE_Signature 結構。COSE 簽章訊息的範例可以在附錄 C.1 中找到。

COSE_Sign 結構是 CBOR 陣列。陣列的欄位依次為：

保護：如第 3 節所述。
未保護：如第 3 節所述。
負載:此欄位包含要簽章的序列化內容。如果訊息中不存在負載，則應用程式需要單獨提供負載。負載包裹在一個 bstr 中，以確保它無需更改即可傳輸。如果負載是單獨運輸的（“分離的內容”），那麼將一個零 CBOR 物件放在該位置，並且應用程式有責任確保在沒有變化的情況下運輸它。

注意：當使用帶有訊息恢復演算法的簽章時（第 8 節），可以恢復的最大位元組數是負載的長度。負載的大小減少將要恢復的位元組數。如果消耗負載的所有位元組，則負載被編碼成零長度二進制字串而不是消失了。

簽章：此欄位是一個簽章陣列。每個簽章都表示為 COSE_Signature 結構。
下面是代表 COSE_Sign 上述文本的 CDDL 片段。

```
COSE_Sign = [
   Headers,
  payload : bstr / nil,
  signatures : [+ COSE_Signature]
]
```

COSE_Signature 結構是 CBOR 陣列。陣列的欄位依次為：

保護：如第 3 節所述。
未保護：如第 3 節所述。
簽章：該欄位包含計算的簽章值。欄位的型態是 bstr。如果簽章值不是 8 位元的倍數，演算法必須指定填充。
下面是代表 COSE_Signature 上述文本的 CDDL 片段。

```
COSE_Signature = [
   Headers,
   signature : bstr
]
```

4.2. 與一位簽章者簽約

當只有一個簽章放在訊息上時，使用 COSE_Sign1 簽章結構。處理內容和簽章的參數放在同一對儲存區中，而不是分開 COSE_Sign。

結構可以編碼為標籤或未標籤，具體取決於它將使用的上下文。標籤的 COSE_Sign1 結構由 CBOR 標籤 18 辨識。表示此結構的 CDDL 片段是

攜帶正文的 CBOR 物件，簽章以及有關正文和簽章的資訊稱為 COSE_Sign1 結構。COSE_Sign1 訊息的範例可以在附錄 C.2 中找到。

COSE_Sign1 結構是 CBOR 陣列。陣列的欄位依次為：

保護：如第 3 節所述。
未保護：如第 3 節所述。
負載：如第 4.1 節所述。
簽章：該欄位包含計算的簽章值。欄位的型態是 bstr。
下面是代表 COSE_Sign1 的上述文本的 CDDL 片段。

```
COSE_Sign1 = [
   Headers,
   payload : bstr / nil,
   signature : bstr
]
```

4.3. 外部提供的資料

COSE 文件中提供的功能之一是應用程式能夠提供要進行身份驗證的其他資料，但這不是 COSE 物件的一部分。通過查看 CoAP 訊息結構[RFC7252]可以看出支援這一點的主要原因，其中存在用於在負載之前攜帶的選項的設施。可以放在此位置的資料範例是 CoAP 編碼或 CoAP 選項。如果資料位於標頭部分，則代理可以使用它來幫助執行其操作。例如，代理可以使用接受選項來確定代理的快取中是否有適當的值。但是發送者可以通過在生成的身份驗證標籤中包含該值來阻止代理更改它將接受的值集合。然而，也可能希望保護這些值，以便如果它們在傳輸中被修改，則可以檢測它們。

本文描述使用外部提供的經過驗證的資料的位元組陣列的過程；但是，建構位元組陣列的方法是應用程式的一個功能。使用此功能的應用程式需要定義如何建構外部提供身份驗證的資料。這種結構需要考慮以下問題：

- 如果包含多個項目，則應用程式需要確保在輸入不同時不會生成相同的位元組字串。這可以通過附加字串'AB'和'CDE'或通過附加字串'ABC'和'DE'來實現。這通常通過使欄位為固定寬度 and/or 將欄位長度編碼為輸出的一部分來解決。使用 CoAP [RFC7252]中的選項作為範例，這些欄位使用 TLV 結構，因此它們可以連接而沒有任何問題。

- 如果包含多個項目，則需要定義項目的訂單。使用 CoAP 中的選項作為範例，應用程式可以聲明欄位將按選項編號排序。

- 應用程式需要確保雙方的位元組流相同。如果保留相同的相對編號，則使用 CoAP 中的選項可能會出現問題。中間節點可以插入或刪除選項，從而更改相對編號的完成方式。應用程式需要指定必須將相對數字重新編碼為僅相對於外部資料中的選項。

4.4. 簽章和驗證過程

為了建立簽章，需要定義明確的位元組流。 Sig_structure 用於建立規範形式。此簽章和驗證過程接收正文資訊（COSE_Sign 或 COSE_Sign1），簽章者資訊（COSE_Signature）和應用程式資料（外部源）。 Sig_structure 是 CBOR 陣列。 Sig_structure 的欄位依次為：

1. 辨識簽章上下文的文本字串。上下文字串為：
   使用 COSE_Signature 結構簽章的”Signature”。
   使用 COSE_Sign1 結構的簽章的”Signature1”。
   用於簽章的 “CounterSignature”使用為計數器簽章屬性。

2. 以 bstr 型態編碼的正文結構的受保護屬性。如果沒有受保護的屬性，則使用長度為零的 bstr。

3. 以 bstr 型態編碼的簽章者結構中的受保護屬性。如果沒有受保護的屬性，則使用長度為零的 bstr。COSE_Sign1 簽章結構省略該欄位。

4. 以 bstr 型態編碼的應用程式中的受保護屬性。如果未提供此欄位，則預設為零長度二進制字串。（有關建構此欄位的應用規則，請參閱第 4.3 節。）

5. 要簽章的負載以 bstr 型態編碼。負載放置在這裡，與其傳輸方式無關。

描述上述文本的 CDDL 片段是：

```
Sig_structure = [
   context : "Signature" / "Signature1" /
   "CounterSignature",
   body_protected : empty_or_serialized_map,
   ? sign_protected : empty_or_serialized_map,
   external_aad : bstr,
   payload : bstr
]
```

如何計算簽章：

1. 建立 Sig_structure 並使用適當的欄位填充它。
2. 使用第 14 節中描述的編碼，通過將 Sig_structure 編碼為位元組字串來建立值 ToBeSigned。
3. 調用簽章建立演算法，傳入 K（要簽章的密鑰），alg（要簽章的演算法）和 ToBeSigned（要簽章的值）。
4. 將生成的簽章值放在陣列的“signature”欄位中。

驗證簽章的步驟如下：

1. 建立 Sig_structure 物件並使用適當的欄位填充它。
2. 使用第 14 節中描述的編碼，通過將 Sig_structure 編碼為位元組字串來建立值 ToBeSigned。

3. 調用簽章驗證演算法，傳入 K（用於驗證的密鑰），alg（用於簽章的算法），ToBeSigned（要簽章的值）和 sig（要驗證的簽章）。

除了執行簽章驗證之外，應用程式還可以執行適當的檢查以確保密鑰與簽章身份正確配對並且在執行動作之前授權簽章身份。

4.5. 計算計數器簽章

計數器簽章提供一種將不同簽章者生成的不同簽章與某個內容相關聯的方法。這通常用於在簽章上提供簽章，允許證明簽章在給定時間存在（即，時間戳）。在本文中，我們允許在更多環境中存在計數器簽章。例如，可以將計數器簽章放在 COSE_Encrypt 物件的不受保護的屬性中。這將允許中間驗證加密的位元組流是否未被修改，無法解密，或者斷言加密的位元組流在給定時間存在或者在路由方面通過它（即，代理簽章）。

簽章上的計數器簽章範例可以在附錄 C.1.3 中找到。加密物件中的計數器簽章範例可以在附錄 C.3.3 中找到。

在不同項目上建立和驗證計數器簽章依賴於物件具有相同結構的事實。元素是一組受保護的屬性，一組不受保護的屬性和一個正文。這意味著可以以統一的方式使用 Sig_structure 來獲得用於處理簽章的位元組流。如果計數器簽章將通過 COSE_Encrypt 結構計算，則正文保護和負載項可以以與 COSE_Sign 結構相同的方式映射到 Sig_structure。

應該注意的是，只有在附錄的簽章演算法（見第 8 節）可用於計數器簽章。這是因為應該能夠處理正文而不必評估計數器簽章，這對於具有訊息恢復的簽章方案是不可能的。

5. 加密物件

COSE 支援兩種不同的加密結構。當不需要接收者結構時使用 COSE_Encrypt0，因為要隱含地使用要使用的密鑰。其餘時間使

用 COSE_Encrypt。這包括多個接收者或使用直接接收者以外的接收者演算法之情況。

5.1. 封裝的 COSE 結構

封裝結構考慮到訊息的一個或多個接收者。關於要在訊息中攜帶的接收者資訊之內容和參數有參數的規定。與內容相關聯的受保護參數由內容加密演算法驗證。與接收者關聯的受保護參數由接收者演算法驗證（當演算法支援時）。關於內容的參數範例是內容的型態和內容加密演算法。有關接收者的參數範例是接收者的密鑰識別符和接收者的加密演算法。

相同的技術和結構用於加密明文和密鑰。這與“加密訊息語法（CMS）”[RFC5652]和“JSON Web 加密（JWE）”[RFC7516]使用的方法不同，其中不同的結構用於內容層和接收層。定義兩種結構：COSE_Encrypt 用於保存加密內容，COSE_recipient 用於保存接收者的已加密密鑰。加密訊息的範例可以在附錄 C.3 中找到。

COSE_Encrypt 結構可以編碼為標籤或未標籤，具體取決於它將使用的上下文。標籤的 COSE_Encrypt 結構由 CBOR 標籤 96 辨識。表示此的 CDDL 片段是：

COSE_Encrypt_Tagged = #6.96(COSE_Encrypt)

COSE_Encrypt 結構是 CBOR 陣列。陣列的欄位依次為：

保護：如第 3 節中所述。
未保護：如第 3 節所述。
密文：該欄位包含編碼為 bstr 的密文。如如果密文為要獨立傳輸關於加密過程的控制信息（即，分離的內容），則將該欄位編碼為零值。
接收者：此欄位包含一組接收者信息結構。接收者信息結構的型態是 COSE_recipient。
與上述文本對應的 CDDL 片段是：

```
COSE_Encrypt = [
   Headers,
   ciphertext : bstr / nil,
   recipients : [+COSE_recipient]
]
```

COSE_recipient 結構是 CBOR 陣列。陣列的欄位依次為：

保護：如第 3 節所述。

未保護：如第 3 節所述。

密文：該欄位包含編碼為 bstr 的已加密密鑰。所有編碼的密鑰都是對稱密鑰;密鑰的二進制值是內容。如果沒有已加密密鑰，則將該欄位編碼為零值。

接收者：此欄位包含一組接收者信息結構。接收者信息結構的型態是 COSE_recipient（可以在附錄 B 中找到此範例）。如果沒有接收者信息結構，則此元素不存在。

與 COSE_recipient 的上述文本對應的 CDDL 片段是：

```
COSE_recipient = [
   Headers,
   ciphertext : bstr / nil,
   ? recipients : [+COSE_recipient]
]
```

5.1.1. 內容密鑰分配方法

密文由加密內容和一個或多個接收者的加密 CEK 組成。使用特定於該接收者的密鑰為每個接收者加密 CEK。此加密的詳細資訊取決於接收方演算法屬於哪個層級。有關每個層級的具體詳細資訊，請參見第 12 節。五種內容密鑰分配方法的簡短摘要如下：

直接：CEK 與辨識出先前分佈的對稱密鑰相同或者從先前分佈的機密中導出。訊息中沒有傳輸 CEK。

對稱密鑰加密鑰（KEK）：CEK 使用先前分佈的對稱 KEK 加密。也稱為密鑰包裝。

密鑰協定：接收者的公鑰和發送者的私鑰用於產生配對機密，金鑰推衍函數（KDF）用於導出金鑰，然後 CEK 是衍生金鑰或由衍生金鑰加密。

密鑰傳輸：CEK 使用接收者的公鑰加密。本文中未定義任何密鑰傳輸演算法。

密碼：CEK 在 KEK 中加密，該密鑰源自密碼。本文中未定義密碼演算法。

5.2. 單一接收者加密

COSE_Encrypt0 加密結構無法指定訊息的接收者。該結構假設物件的接收者已經知道要使用的密鑰的身份以便解密訊息。如果需要向接收者辨識密鑰，則應該使用封裝結構。

加密訊息的範例可以在附錄 C.3 中找到。
COSE_Encrypt0 結構可以編碼為標籤或未標籤，具體取決於它將使用的上下文。標籤的 COSE_Encrypt0 結構由 CBOR 標籤 16 辨識。表示此的 CDDL 片段是：

COSE_Encrypt0_Tagged = #6.16(COSE_Encrypt0)

COSE_Encrypt0 結構是 CBOR 陣列。陣列的欄位依次為：

保護：如第 3 節中所述。
未保護：如第 3 節所述。
密文：如第 5.1 節中所述。

與上述文本對應的 COSE_Encrypt0 的 CDDL 片段是：

```
COSE_Encrypt0 = [
   Headers,
   ciphertext : bstr / nil,
]
```

5.3. 如何加密和解密 AEAD 演算法

AEAD 演算法的加密演算法相當簡單。第一步是為經過身份驗證的資料結構建立一致的位元組流。為此，我們使用 Enc_structure。Enc_structure 是一個 CBOR 陣列。 Enc_structure 的欄位依次為：

1. 標識經過身份驗證的資料結構的上下文的文本字串。上下文字串是：
   “Encrypt0”用於 COSE_Encrypt0 資料結構的內容加密。
   “Encrypt”用於 COSE_Encrypt 資料結構的第一層（即，用於內容加密）。“Enc_Recipient”用於將接收者編碼放置在

COSE_Encrypt 資料結構中。“Mac_Recipient”用於將接收者編碼放置在 MACed 訊息結構中。

“Rec_Recipient” 用於將接收者編碼放置在接收者結構中。

2. 以 bstr 型態編碼的正文結構的受保護屬性。如果沒有受保護的屬性，則使用長度為零的 bstr。
3. 以 bstr 型態編碼的應用程式的受保護屬性。如果未提供此欄位，則預設為零長度的 bstr。（有關建構此欄位的應用規則，請參閱第 4.3 節。）

描述上述文本的 CDDL 片段是：

```
Enc_structure = [
    context : "Encrypt" / "Encrypt0" / "Enc_Recipient" /
        "Mac_Recipient" / "Rec_Recipient",
    protected : empty_or_serialized_map,
    external_aad : bstr
]
```

如何加密訊息：

1. 建立 Enc_structure 並使用適當的欄位填充它。
2. 使用第 14 節中描述的編碼將 Enc_structure 編碼為位元組流（額外驗證資料（AAD））。
3. 確定加密密鑰（K）。此步驟取決於所使用的接收者演算法類別。對於：

   無接收者：要使用的密鑰由當前層的演算法和密鑰決定。範例是密鑰傳送密鑰（第 12.3 節），密鑰包裝密鑰（第 12.2.1 節）或預先共用密碼。

   直接加密和直接密鑰協定：密鑰由接收者結構中的密鑰和演算法確定。加密演算法和要使用的密鑰的大小是用於接收者的 KDF 的輸入。（對於直接，KDF 可以被認為是恆等運算。）這些演算法的例子可以在第 12.1.2 節和第 12.4.1 節中找到。

   其他：密鑰隨機或偽隨機產生。
4. 使用 K（加密密鑰），P（明文）和 AAD 調用加密演算法。將回傳的密文放入結構的“密文”欄位中。

5. 對於訊息的接收者，使用 K（加密密鑰）作為明文，為該接收者遞迴地執行加密演算法。

如何解密訊息：

1. 建立 Enc_structure 並使用適當的欄位填充它。
2. 使用第 14 節中描述的編碼將 Enc_structure 編碼為位元組流（AAD）。
3. 確定解密密鑰。此步驟取決於所使用的接收者演算法類別。對於：

   無接收者：要使用的密鑰由當前層的演算法和密鑰決定。範例是密鑰傳送密鑰（第 12.3 節），密鑰包裝密鑰（第 12.2.1 節）或預先共用密碼。

   直接加密和直接密鑰協定：密鑰由接收者結構中的密鑰和演算法確定。加密演算法和要使用的密鑰的大小是用於接收者的 KDF 的輸入。（對於直接，KDF 可以被認為是恆等運算。）這些演算法的例子可以在第 12.1.2 節和第 12.4.1 節中找到。

   其他：密鑰是通過解碼和解密其中一個接收者結構來確定的。
4. 使用 K（要使用的解密密鑰），C（密文）和 AAD 調用解密演算法。

5.4. 如何加密和解密 AE 演算法

如何加密訊息：

1. 驗證“protected”欄位是否為空。
2. 驗證是否沒有為此操作提供外部附加的身份驗證資料。
3. 確定加密密鑰。此步驟取決於所使用的接收者演算法類別。對於：

   無接收者：要使用的密鑰由當前層的演算法和密鑰決定。範例是密鑰傳輸密鑰（第 12.3 節），密鑰包裝密鑰（第 12.2.1 節）或預先共用密碼。

直接加密和直接密鑰協定：密鑰由接收者結構中的密鑰和演算法確定。加密演算法和要使用的密鑰的大小是用於接收者的 KDF 的輸入。（對於直接，KDF 可以被認為是恆等運算。）這些演算法的例子可以在第 12.1.2 節和第 12.4.1 節中找到。

其他：密鑰是隨機產生。

4. 使用 K（要使用的加密密鑰）和 P（明文）調用加密演算法。將返回的密文放入結構的“密文”欄位中。
5. 對於訊息的接收者，使用 K（加密密鑰）作為明文，為該接收者遞迴地執行加密演算法。

如何解密訊息：

1. 驗證“protected”欄位是否為空。
2. 驗證是否沒有為此操作提供外部額外驗證資料。
3. 確定解密密鑰。此步驟取決於所使用的接收者演算法類別。

對於：

無接收者：要使用的密鑰由當前層的演算法和密鑰決定。範例是密鑰傳輸密鑰（第 12.3 節），密鑰包裝密鑰（第 12.2.1 節）或預先共用密碼。

直接加密和直接密鑰協定：密鑰由接收者結構中的密鑰和演算法確定。

加密演算法和要使用的密鑰的大小是用於接收者的 KDF 的輸入。（對於直接，KDF 可以被視為是恆等運算。）這些演算法的例子可以在第 12.1.2 節和第 12.4.1 節中找到。

其他：密鑰是通過解碼和解密其中一個接收者結構來確定的。

4. 使用 K（要使用的解密密鑰）和 C（密文）調用解密演算法。

6. MAC 物件

COSE 支援兩種不同的 MAC 結構。當不需要接收者結構時使用 COSE_MAC0，因為隱含地知道要使用的密鑰。 COSE_MAC 用於所有其他情況。這些包括對多個接收者的要求，密鑰未知，以及除直接之外的接收者演算法。

在本節中，我們將描述在 COSE 中進行 MAC 身份驗證時要使用的結構和方法。該文允許使用允許加密的所有相同類別的接收者演算法。

使用 MAC 操作時，有兩種模式可供使用。第一個是只檢查自計算 MAC 以來內容是否未更改。任何類別的接收者演算法都可用於此方法。第二種模式是檢查自計算MAC以來內容是否未更改，並使用接收者演算法驗證發送者是誰。支援此功能的接收者演算法類別是使用預先共用密碼或執行靜態 - 靜態（SS）密鑰協定的階級（不使用密鑰包裝步驟）。在這兩種情況下，可以驗證建立和發送訊息 MAC 的實體。（發送者的這種知識假設只涉及兩方並且您沒有將訊息發送給自己。）可以使用兩種 MAC 訊息結構獲得原始屬性。

6.1. 接收者的 MACed 訊息

多個接收者 MACed 訊息使用兩種結構：本節中定義的用於承載主體的 COSE_Mac 結構和用於 MAC 計算的 COSE_recipient 結構（第 5.1 節）。 MACed 訊息的範例可以在附錄 C.5 中找到。

MAC結構可以編碼為標籤或未標籤，具體取決於它將使用的上下文。標籤的COSE_Mac結構由CBOR標籤97辨識。表示此的CDDL片段是：

```
COSE_Mac_Tagged = #6.97(COSE_Mac)
```

COSE_Mac 結構是 CBOR 陣列。陣列的欄位依次為：

保護：如第 3 節中所述。
未保護：如第 3 節所述。
負載：該欄位包含要 MACed 的序列化內容。如果訊息中不存在負載，則應用程式需要單獨提供負載。負載包裹在一個 bstr 中，以確保它無需更改即可傳輸。如果負載是單獨運輸的（即，分離的內容），那麼在該位置放置一個零 CBOR 值，並且應用程式有責任確保在沒有變化的情況下運輸它。
標籤：該欄位包含 MAC 值。

接收者：如第 5.1 節中所述的。

下面是代表 COSE_Mac 的上述文本的 CDDL 片段。

```
COSE_Mac = [
   Headers,
   payload : bstr / nil,
   tag : bstr,
   recipients :[+COSE_recipient]
]
```

6.2. 具有隱性鍵值的 MACed 訊息

在本節中，我們將描述在對隱性知道接收者的情況進行 MAC 身份驗證時要使用的結構和方法。

MACed 訊息使用本節中定義的 COSE_Mac0 結構來攜帶正文。帶有隱性鍵值的 MAC 訊息範例可以在附錄 C.6 中找到。

MAC 結構可以編碼為標籤或未標籤，具體取決於它將使用的上下文。標籤的 COSE_Mac0 結構由 CBOR 標籤 17 辨識。表示此結構的 CDDL 片段是：

COSE_Mac0_Tagged = #6.17(COSE_Mac0)

COSE_Mac0 結構是 CBOR 陣列。陣列的欄位依次為：

保護：如第 3 節所述。
未保護：如第 3 節所述。
負載：如第 6.1 節所述。
標籤：該欄位包含 MAC 值。

與上述文本對應的 CDDL 片段是：

```
COSE_Mac0 = [
   Headers,
   payload : bstr / nil,
   tag : bstr,
]
```

6.3. 如何計算和驗證 MAC

為了獲得要認證的資料的一致編碼，MAC_structure 用於具有規範形式。 MAC_structure 是 CBOR 陣列。

MAC_structure 的欄位依次為：

1. 辨識正在編碼的結構的文本字串。該字串是 COSE_Mac 結構的“MAC”。該字串是 COSE_Mac0 結構的“MAC0”
2. 來自 COSE_MAC 結構的受保護屬性。如果沒有受保護的屬性，則使用零長度的 bstr。
3. 應用程式的受保護屬性編碼為 bstr 型態。如果未提供此欄位，則預設為零長度二進制字串。（有關建構此欄位的應用規則，請參閱第 4.3 節。）
4. 要以 bstr 型態進行 MAC 編碼的負載。負載放置在這裡，與其運輸方式無關。

與上述文本對應的 CDDL 片段是：

```
MAC_structure = [
     context : "MAC" / "MAC0",
     protected : empty_or_serialized_map,
     external_aad : bstr,
     payload : bstr
]
```

計算 MAC 的步驟如下：

1. 建立 MAC_structure 並使用適當的欄位填充它。
2. 通過使用第 14 節中描述的編碼將 MAC_structure 編碼為位元組流來建立值 ToBeMaced。
3. 調用 MAC 建立演算法，傳入 K（要使用的密鑰），alg（演算法到 MAC）和 ToBeMaced（用於計算 MAC 的值）。
4. 將生成的 MAC 放在 COSE_Mac 或 COSE_Mac0 結構的“tag”欄位中。
5. 為每個訊息接收者加密和編碼 MAC 密鑰。

驗證 MAC 的步驟如下：

1. 建立 MAC_structure 物件並使用適當的欄位填充它。

2. 通過使用第 14 節中描述的編碼將 MAC_structure 編碼為位元組流來建立值 ToBeMaced。
3. 從訊息其中的一個接收者獲取加密密鑰。
4. 調用 MAC 建立演算法，傳入 K（要使用的密鑰），alg（演算法到 MAC）和 ToBeMaced（用於計算 MAC 的值）。
5. 將 MAC 值與 COSE_Mac 或 COSE_Mac0 結構的“tag”欄位進行比較。

7. 密鑰物件

COSE 密鑰結構建構在 CBOR 映射物件上。可以在 IANA“COSE 密鑰常見參數”註冊表（第 16.5 節）中找到可以出現在 COSE 密鑰中的一組常見參數。可以在 IANA“COSE 密鑰型態參數”註冊表（第 16.6 節）中找到為特定密鑰型態定義的其他參數。

COSE 密鑰集使用 CBOR 陣列物件作為其基礎型態。陣列元素的值是 COSE 密鑰。COSE 密鑰集必須在陣列中至少有一個元素。COSE 密鑰集的範例可以在附錄 C.7 中找到。

COSE 密鑰集中的每個元素必須獨立處理。如果 COSE 密鑰集中的一個元素格式錯誤或使用應用程式無法理解的密鑰，則忽略該密鑰並正常處理其他密鑰。

元素“kty”是 COSE_Key 映射中的必需元素。

描述 COSE_Key 和 COSE_KeySet 的 CDDL 語法是：

```
COSE_Key = {
   1 => tstr / int,          ; kty
   ? 2 => bstr,              ; kid
   ? 3 => tstr / int,        ; alg
   ? 4 => [+ (tstr / int) ], ; key_ops
   ? 5 => bstr,              ; Base IV
   * label => values
}

COSE_KeySet = [+COSE_Key]
```

7.1. COSE 鍵值常用參數

本文為 COSE Key 物件定義一組常見參數。表 3 提供本節中定義的參數的摘要。還有為特定鍵值型態定義的參數。特定鍵值型態的參數可以在第 13 節中找到。

Name	Label	CBOR Type	Value Registry	Description
kty	1	tstr / int	COSE Key Common Parameters	Identification of the key type
kid	2	bstr		Key identification value -- match to kid in message
alg	3	tstr / int	COSE Algorithms	Key usage restriction to this algorithm
key_ops	4	[+ (tstr/int)]		Restrict set of permissible operations
Base IV	5	bstr		Base IV to be xor-ed with Partial IVs

表 3：映射鍵值標籤

kty：此參數用於標識此結構的密鑰家族，因此，用於標識要找到的鍵值型態特定參數的集合。可以在表 21 中找到本文中定義的值集合。此參數必須存在於密鑰物件中。實現必須驗證密鑰型態是否適合正在處理的演算法。密鑰型態必須作為信任決策過程的一部分包含在內。

alg：此參數用於限制與密鑰一起使用的演算法。如果密鑰結構中存在此參數，則應用程式必須驗證此演算法是否與正在使用密鑰的演算法匹配。如果演算法不匹配，則不得使用此密鑰物件來執行加密操作。注意，相同的密鑰可以在不同的密鑰結構中，指定不同的或不指定演算法；但是，這被認為是一種不良的安全作法。

kid：此參數用於為密鑰提供識別符。識別符不是結構化的，可以是從用戶提供的字串到在密鑰的公共部分上計算的值的任何內容。此欄位用於匹配訊息中的“kid”參數，以便過濾掉需要檢查的密鑰集。

key_ops：定義此參數以限制密鑰用於這套操作。該欄位的值是表4中的值陣列。演算法定義允許出現並且是特定操作所需的鍵操作的值。這一連串的值與[RFC7517]和[W3C.WebCrypto]中的值匹配。

基礎IV（Base IV）：該參數被定義為攜帶IV的基礎部分。它設計用於3.1節中定義的部分IV標頭參數。此欄位提供將部分IV與密鑰相關聯的功能，然後使用部分IV將每個訊息修改。

在應用程式中使用基礎IV時需要特別小心。如果兩次使用相同的IV，許多加密演算法會失去安全性。
如果為每個發送者導出不同的密鑰，則使用從零開始的具有部分IV的相同基礎IV可能確保對於單個密鑰不會使用IV兩次。如果為每個發送者導出不同的密鑰，則從相同的基礎IV開始可能滿足此條件。如果相同的密鑰用於多個發送者，則應用程式需要提供在發送者之間劃分IV空間的方法。這可以通過提供不同的基點來開始或從不同的部分IV開始並限制在重新加密之前發送的訊息的數量來完成。

Name	Value	Description
sign	1	The key is used to create signatures. Requires private key fields.
verify	2	The key is used for verification of signatures.
encrypt	3	The key is used for key transport encryption.
decrypt	4	The key is used for key transport decryption. Requires private key fields.
wrap key	5	The key is used for key wrap encryption.
unwrap key	6	The key is used for key wrap decryption. Requires private key fields.
derive key	7	The key is used for deriving keys. Requires private key fields.
derive bits	8	The key is used for deriving bits not to be used as a key. Requires private key fields.
MAC create	9	The key is used for creating MACs.
MAC verify	10	The key is used for validating MACs.

表 4：鍵值操作值

8. 簽章演算法

有兩種簽章演算法方案。第一個是附錄簽章。在該方案中，處理訊息內容並生成簽章；簽章稱為附錄。這是像 ECDSA 和 RSA 概率簽章方案（RSASSA-PSS）之類的演算法所使用的方案。（事實上，RSASSA-PSS 中的 SSA 代表附錄中的簽章方案。）

該方案的簽章功能是：

```
signature = Sign(message content, key)
valid = Verification(message content, key, signature)
```

第二種方案是具有訊息恢復的簽章（這種演算法的一個例子是 [PVSig]）。在此方案中，將處理訊息內容，但其中一部分將包含在簽章中。將訊息內容的位元組移動到簽章中允許更小的簽章；簽章大小仍然可能很大，但訊息內容已縮小。這對實現這些演算法的系統和使用它們的應用程式有影響。首先，訊息內容在簽章驗證之後才能完全可用。在此之前，簽章中包含的訊息部分是不可恢復的。第二個是簽章強度的安全性分析非常基於訊息內容的結構。高度可預測的訊息需要額外的隨機性作為簽章過程的一部

分。在最壞的情況下，它與使用附錄進行簽章相同。最後，在將多個簽章應用於訊息的情況下，將要求所有簽章演算法消耗相同數量的訊息內容位元組。這意味著不支援在單個訊息中混合不同方案，並且如果使用恢復簽章方案，則所有簽章都需要消耗相同數量的內容。

該方案的簽章功能是：

```
signature = Sign(message content, key)
valid = Verification(message content, key, signature)
```

簽章演算法與 COSE_Signature 和 COSE_Sign1 結構一起使用。目前，只有帶附錄的簽章被定義為與 COSE 一起使用；然而，由於可能的有效尺寸減小，在使用具有訊息恢復演算法的簽章方面表達相當大的興趣。實現將需要牢記這一點，以便以後可能的集成。

8.1. ECDSA

ECDSA [DSS]使用 ECC 定義簽章演算法。實現 SHOULD 使用 ECDSA 的確定性版本，例如[RFC6979]中定義的版本。確定性簽章演算法的使用允許系統避免依賴於隨機數生成器以避免生成相同的‘k’值（每一訊息隨機值）。可以攻擊‘k’值的偏差生成，並且該值的碰撞導致洩漏密鑰。它還允許對簽章演算法進行確定性測試。確定性 ECDSA 的使用不會減少在建立私鑰時有良好隨機數產生。

ECDSA 簽章演算法使用雜湊函數（h）進行參數化。如果雜湊函數輸出的長度大於密鑰組，則使用雜湊輸出的最左邊的位元組。

本文中定義的演算法可以在表 5 中找到。

Name	Value	Hash	Description
ES256	-7	SHA-256	ECDSA w/ SHA-256
ES384	-35	SHA-384	ECDSA w/ SHA-384
ES512	-36	SHA-512	ECDSA w/ SHA-512

表 5：ECDSA 演算法的值

本文將 ECDSA 定義為僅適用於曲線 P-256，P-384 和 P-521。本文要求使用 ‘EC2’（2 坐標橢圓曲線）鍵值型態對曲線進行編碼。在建立和驗證簽章時，實現需要檢查密鑰型態和曲線是否正確。其他文件可以定義它以便將來與其他曲線和點一起使用。

為了提高可交互運作性，建議 SHA-256 僅用於曲線 P-256，SHA-384 僅用於曲線 P-384，SHA-512 用於曲線 P-521。這與 [RFC5480]第 4 節中的建議一致。

簽章演算法產生一對整數（R，S）。這些整數的長度與用於簽章過程的密鑰長度相同。通過將整數轉換為與密鑰大小相同長度的位元組串來編碼簽章。長度被四捨五入到最接近的位元組，並用零位左填充以獲得正確的長度。然後將這兩個整數連接在一起形成一個位元組字串，這是生成的簽章。
使用[RFC8017]中定義的功能，簽章為：

```
Signature = I2OSP(R, n) | I2OSP(S, n)
where n = ceiling(key_length / 8)
```

對此演算法使用 COSE 密鑰時，將進行以下檢查：

- 必須存在 ‘kty’ 欄位，它必須是 ‘EC2’ 。
- 如果存在 ‘alg’ 欄位，它必須與正在使用的 ECDSA 簽章演算法匹配。
- 如果存在 ‘key_ops’ 欄位，則在建立 ECDSA 簽章時必須包含 ‘sign’ 。
- 如果存在 ‘key_ops’ 欄位，則在驗證 ECDSA 簽章時必須包括 ‘verify’ 。

8.1.1. 安全考慮因素

簽章的安全強度不大於與密鑰的位元長度和雜湊函數的安全強度相關的安全強度的最小值。

注意:即使存在良好的隨機數生成,使用此技術也是一個好主意。這樣的做法既降低了在兩個簽章操作中具有相同“k”值的可能性，又允許可重現的簽章值，這有助於測試。
理論上可以針對 ECDSA 簽章演算法承載兩種替代攻擊。

- 更改用於驗證簽章的曲線：如果更改用於驗證簽章的曲線，則可能會有兩條具有相同簽章的訊息，每條訊息在不同的曲線下計算。新曲線的唯一要求是它的順序與舊曲線相同，並且客戶端可以接受。一個例子是從使用曲線 secp256r1（又名 P-256）改為使用 secp256k1。（兩者都是 256 位曲線。）我們目前沒有辦法處理這種攻擊版本，除了限制可以使用的整個曲線集。
- 更改用於驗證簽章的雜湊函數：如果有一個具有相同長度的兩個不同雜湊函數或者可以截斷雜湊函數，則可能會發現雜湊函數之間的衝突而不是單個雜湊函數內的衝突(對於例如，將 SHA-512 截斷為 256 位元可能會與 SHA-256 位元雜湊值衝突）。由於雜湊演算法是簽章演算法識別符的一部分，因此通過在受保護的標頭中包括簽章演算法識別符來減輕該攻擊。

8.2. Edwards 曲線數位簽章演算法（EdDSAs）

[RFC8032]描述橢圓曲線簽章方案 Edwards 曲線數位簽章演算法（EdDSA）。在該文中，使用 edwards25519 和 edwards448 曲線的參數來實體化簽章演算法。該文還描述 EdDSA 演算法的兩種變體：純 EdDSA，其中在簽章之前沒有對內容應用雜湊函數，以及雜湊 EdDSA,其中在簽章之前將雜湊函數應用於內容並且該雜湊函數的結果被簽章。對於 EdDSA，要簽章的內容（訊息或預先雜湊值）在簽章演算法內處理兩次。與 COSE 一起使用時，僅使用純 EdDSA 版本。這是因為不期望需要非常大的內容，並且基於訊息結構的佈置，整個訊息將需要保存在存儲器中以便建立或驗證簽章。這意味著似乎不需要能夠對雜湊進行區塊更新，然後從存儲器中消除訊息。應用程式可以通過將訊息內容定義為雜湊值並將 COSE 物件（具有雜湊值）和內容作為單獨項目進行傳輸來提供相同的功能。

本文中定義的演算法可以在表 6 中找到。定義單一簽章演算法，可以用於多條曲線。

Name	Value	Description
EdDSA	-8	EdDSA

表 6：EdDSA 演算法的值

[RFC8032]描述對簽章值進行編碼的方法。

對此演算法使用 COSE 密鑰時，將進行以下檢查：

- 必須存在'kty'欄位，並且它必須是'OKP'（位元組密鑰對）。
- 必須存在'crv'欄位，它必須是為此簽章演算法定義的曲線。
- 如果'alg'欄位存在，它必須匹配'EdDSA'。
- 如果存在'key_ops'欄位，則在建立 EdDSA 簽章時必須包含'sign'。
- 如果存在'key_ops'欄位，則在驗證 EdDSA 簽章時必須包括'verify'。

8.2.1. 安全考慮因素

在查看 EdDSA 和 Diffie-Hellman 橢圓曲線（ECDH）時，如何計算公共值是不一樣的；因此，它們不應與其他演算法一起使用。

如果執行批簽章驗證，則需要具有良好種子的加密隨機數生成器。簽章和非批量簽章驗證是確定性操作，不需要任何型態的隨機數。

9. 訊息鑑別碼（MAC）演算法

訊息鑑別碼（MAC）提供資料認證和完整性保護。它們提供無資料或非常有限的資料產生。例如，MAC 可用於向第三方證明發送者的身份。

MAC 使用與附錄演算法簽章相同的方案。處理訊息內容並產生認證編碼。驗證編碼通常稱為標籤。

MAC 功能是：

```
tag = MAC_Create(message content, key)
valid = MAC_Verify(message content, key, tag)
```

MAC演算法可以基於分塊加密演算法（即，AES-MAC）或雜湊演算法（即，基於雜湊訊息鑑別碼（HMAC））。本文定義使用這些結構中的每一種的MAC演算法。

MAC演算法用於COSE_Mac和COSE_Mac0結構。

9.1. 雜湊訊息鑑別碼（HMAC）

HMAC [RFC2104] [RFC4231]旨在處理長度擴充攻擊。該演算法還被設計為允許直接插入新的雜湊演算法而無需改變雜湊函數。HMAC 設計過程已被證明是可靠的，因為 MD5 等雜湊演算法的安全性隨著時間的推移而降低；HMAC 與 MD5 結合的安全性尚未證明被破解[RFC6151]。

HMAC 演算法通過內部和外部填充，雜湊函數（h）和認證標籤值長度來參數化。對於此規範，內部和外部填充固定為[RFC2104]中設置的值。認證標籤的長度對應於產生偽造的難度。為了在受限環境中使用，我們定義一組被截斷的 HMAC 演算法。

目前沒有截斷的已知問題；但是，訊息標籤的安全強度相應地降低強度。截斷時，保留並發送最左邊的標籤長度位元。

本文中定義的演算法可以在表 7 中找到。

Name	Value	Hash	Tag Length	Description
HMAC 256/64	4	SHA-256	64	HMAC w/ SHA-256 truncated to 64 bits
HMAC 256/256	5	SHA-256	256	HMAC w/ SHA-256
HMAC 384/384	6	SHA-384	384	HMAC w/ SHA-384
HMAC 512/512	7	SHA-512	512	HMAC w/ SHA-512

表 7：HMAC 演算法的值

一些接收者演算法攜帶密鑰，而其他接收者演算法從隱密資料導出密鑰。對於那些攜帶密鑰的演算法（例如 AES 密鑰包），HMAC 密鑰的大小應該與底層雜湊函數的大小相同。對於那些導出密鑰的演算法（例如 ECDH），衍生密鑰必須與底層雜湊函數的大小相同。

對此演算法使用 COSE 密鑰時，將進行以下檢查：

- 必須存在'kty'欄位，它必須是'對稱'。
- 如果存在'alg'欄位，它必須與正在使用的 HMAC 演算法匹配。
- 如果存在'key_ops'欄位，則在建立 HMAC 認證標籤時必須包括'MAC create'。
- 如果存在'key_ops'欄位，則在驗證 HMAC 認證標籤時必須包括'MAC verify'。

建立和驗證 MAC 值的實現必須驗證密鑰型態，密鑰長度和演算法是否正確並適用於所涉及的實體。

9.1.1. 安全考慮因素

事實證明，即使與弱式雜湊演算法一起使用，HMAC 也能抵抗攻擊。目前最著名的攻擊是蠻力破解密鑰。這意味著密鑰大小將直接與 HMAC 操作的安全性相關。

9.2. AES 訊息驗證碼（AES-CBC-MAC）

AES-CBC-MAC 在[MAC]中定義。（請注意，這與基於 AES 密碼的訊息驗證編碼（AES-CMAC）[RFC4493]的演算法不同。）

AES-CBC-MAC 通過密鑰長度，認證標籤長度和使用的 IV 來參數化。對於所有這些演算法，IV 固定為全零。我們為各種密鑰長度和標籤長度提供一系列演算法。本文中定義的演算法見表 8。

Name	Value	Key Length	Tag Length	Description
AES-MAC 128/64	14	128	64	AES-MAC 128-bit key, 64-bit tag
AES-MAC 256/64	15	256	64	AES-MAC 256-bit key, 64-bit tag
AES-MAC 128/128	25	128	128	AES-MAC 128-bit key, 128-bit tag
AES-MAC 256/128	26	256	128	AES-MAC 256-bit key, 128-bit tag

表 8：AES-MAC 演算法的值

密鑰可以從密鑰結構或從接收者結構獲得。建立和驗證 MAC 值的實現必須驗證密鑰型態，密鑰長度和演算法是否正確並適用於所涉及的實體。

對此演算法使用 COSE 密鑰時，將進行以下檢查：

- 必須存在'kty'欄位，它必須是'對稱'。
- 如果存在'alg'欄位，它必須與正在使用的 AES-MAC 演算法匹配。
- 如果存在'key_ops'欄位，則在建立 AES-MAC 認證標籤時必須包括'MAC create'。
- 如果存在'key_ops'欄位，則在驗證 AES-MAC 認證標籤時必須包括'MAC verify'。

9.2.1. 安全考慮因素

針對需要考慮的密碼區塊鏈結-訊息鑑別碼（CBC-MAC）存在許多攻擊。

- 單一密鑰只能用於固定和已知長度的訊息。如果不是這種情況，攻擊者將能夠生成帶有兩個訊息和標籤對的有效標籤的訊息。這可以通過對不同長度的訊息使用不同的密鑰來解決。當前結構緩解了這個問題，因為構建並簽署包含長度的特定編碼結構。（CMAC 也解決了這個問題。）
- 密碼區塊鏈結（CBC）模式，如果同一密鑰用於加密和身份驗證操作，則攻擊者可以使用有效的驗證碼生成訊息。
- 如果可以修改 IV，則可以偽造訊息。通過將 IV 固定為全零來解決此問題。

10. 內容加密演算法

內容加密演算法使用對稱密鑰為潛在的大資料區塊提供資料機密性。它們為加密的資料提供完整性；但是，它們提供的資料來源非常有限或沒有。（例如，不能用於向第三方證明發送者的身份。）提供資料起源的能力與獲得 CEK 的方式相關聯。

COSE 將一組合法的內容加密演算法限制為支援對內容和其他資料進行身份驗證的演算法。加密過程將生成某種型態的認證值，但就演算法定義而言，該值可以是顯性的或隱性的。為簡單起見，通常將認證編碼定義為附加到密文流。加密功能是：
大多數 AEAD 演算法在邏輯上被定義為僅在解密有效時才返回訊息內容。許多但並非所有實現都遵循此約定。如果解密未驗證，則不得使用訊息內容。

這些演算法用於 COSE_Encrypt 和 COSE_Encrypt0。

10.1. AES GCM

Galois / Counter Mode（GCM）模式是[AES-GCM]中定義的通用認證加密區塊密碼模式。GCM 模式與 AES 區塊加密演算法結合使用以定義 AEAD 密碼。

GCM 模式通過驗證標籤的大小和隨機數的大小進行參數化。該文將隨機數的大小固定為 96 位元。驗證標籤的大小僅限於一小組值。但是，對於本文，驗證標籤的大小固定為 128 位元。

本文中定義的演算法集如表 9 所示。

Name	Value	Description
A128GCM	1	AES-GCM mode w/ 128-bit key, 128-bit tag
A192GCM	2	AES-GCM mode w/ 192-bit key, 128-bit tag
A256GCM	3	AES-GCM mode w/ 256-bit key, 128-bit tag

表 9：AES-GCM 演算法的值

密鑰可以從密鑰結構或從接收者結構獲得。實現加密和解密必須驗證密鑰型態，密鑰長度和演算法是否正確並適用於所涉及的實體。

對此演算法使用 COSE 密鑰時，將進行以下檢查：

- 必須存在'kty'欄位，它必須是'對稱'。
- 如果存在'alg'欄位，它必須與正在使用的 AES-GCM 演算法匹配。
- 如果存在'key_ops'欄位，則在加密時必須包括'encrypt'或'wrap key'。
- 如果存在'key_ops'欄位，則在解密時必須包括'decrypt'或'unwrap key'。

10.1.1. 安全考慮因素

使用 AES-GCM 時，必須強制執行以下限制：

- 密鑰和隨機數對必須對每個加密的訊息都是唯一的。
- 為單一密鑰加密的資料總量不得超過 2^39 - 256 位元。只有在預計可能超出的環境中才需要進行顯性檢查。

考慮支援較小的標籤值；受約束的團體希望標籤大小在 64 位元範圍內。這樣做會徹底改變最大訊息大小（通常不是問題）和密

鑰可以使用的次數。鑑於具有 CBC-MAC（CCM）的計數器是受約束環境的常用模式，因此不支援受限模式。

10.2. AES CCM

CCM 是[RFC3610]中定義的通用認證加密區塊密碼模式。CCM 模式與 AES 區塊加密演算法相結合，以定義在受約束設備中使用的常用內容加密演算法。

CCM 模式有兩個參數選擇。第一種選擇是 M，即認證欄位的大小。M 值的選擇涉及訊息增長（來自標籤）與攻擊者無法不可檢測地修改訊息的概率之間的權衡。第二個選擇是 L，長度欄位的大小。此值需要在最大訊息大小和隨機數大小之間進行權衡。

不幸的是，CCM 規範將 L 和 M 指定為位元組數而不是位元數。這導致可能的誤解，其中 AES-CCM-8 經常用於 CCM 模式的版本，其中認證的大小是 64 位元而不是 8 位元。傳統上將這些值指定為位元計數而不是位元組計數。本文將遵循使用位元計數的慣例，以便更容易地比較本文中提供的不同演算法。

我們在 L 和 M 的值中定義本文中的演算法矩陣。受限的設備通常在使用短訊息並希望避免進行接收者特定加密操作的情況下運行。這有利於較小的 L 和 M 值。較少受限的設備將希望能夠使用更大的訊息並且更願意為每個操作生成新的密鑰。這有利於更大的 L 和 M 值。

以下值用於 L：

16 位元（2）：這將訊息的長度限制為 2^16 位元組（64 KiB）。這對於受約束世界中的訊息來說足夠長。 隨機數長度為 13 個位元組，允許隨機數的 2^（13 * 8）個可能值不重複

64 位元（8）：這將訊息限制為 2^64 位元組的長度。隨機數長度為 7 個位元組，允許隨機數的 2^56 個可能值不重複

以下值用於 M：

64 位元（8）：這會生成 64 位元驗證標籤。這意味著修改後的訊息將有 1 到 2 ^ 64 的可能性會進行驗證。

128 位元（16）：這會生成 128 位元驗證標籤。這意味著修改後的訊息將有 1 到 2 ^ 128 的可能性會進行驗證。

Name	Value	L	M	k	Description
AES-CCM-16-64-128	10	16	64	128	AES-CCM mode 128-bit key, 64-bit tag, 13-byte nonce
AES-CCM-16-64-256	11	16	64	256	AES-CCM mode 256-bit key, 64-bit tag, 13-byte nonce
AES-CCM-64-64-128	12	64	64	128	AES-CCM mode 128-bit key, 64-bit tag, 7-byte nonce
AES-CCM-64-64-256	13	64	64	256	AES-CCM mode 256-bit key, 64-bit tag, 7-byte nonce
AES-CCM-16-128-128	30	16	128	128	AES-CCM mode 128-bit key, 128-bit tag, 13-byte nonce
AES-CCM-16-128-256	31	16	128	256	AES-CCM mode 256-bit key, 128-bit tag, 13-byte nonce
AES-CCM-64-128-128	32	64	128	128	AES-CCM mode 128-bit key, 128-bit tag, 7-byte nonce
AES-CCM-64-128-256	33	64	128	256	AES-CCM mode 256-bit key, 128-bit tag, 7-byte nonce

表 10：AES-CCM 演算法的值

密鑰可以從密鑰結構或從接收者結構獲得。實現加密和解密必須驗證密鑰型態，密鑰長度和演算法是否正確並適用於所涉及的實體。

對此演算法使用 COSE 密鑰時，將進行以下檢查：

- 必須存在'kty'欄位，它必須是'對稱'。

- 如果存在'alg'欄位，它必須與正在使用的 AES-CCM 演算法匹配。
- 如果存在'key_ops'欄位，則在加密時必須包括'encrypt'或'wrap key'。
- 如果存在'key_ops'欄位，則在解密時必須包括'decrypt'或'unwrap key'。

10.2.1. 安全考慮因素

使用 AES-CCM 時，必須強制執行以下限制：

- 密鑰和隨機數對必須對每個加密的訊息都是唯一的。請注意，L 的值會影響唯一的隨機數。
- 使用 AES 分組密碼的總次數不得超過 2^61 次操作。此限制是區塊密碼用於計算 MAC 值和執行流加密操作的總和。只有在預計可能超出的環境中才需要進行顯性檢查。

[RFC3610]另外提出另一個注意事項。在明文的某些部分是高度可預測的情況下，可以對演算法進行預先計算攻擊。這會將密鑰大小的安全性降低一半。處理此攻擊的方法包括向隨機數值添加隨機部分和/或增加使用的密鑰大小。將一部分隨機數用於隨機值將減少單個密鑰可用於的訊息數。增加密鑰大小可能需要受限制設備中的更多資源。有關更多信息，請參見[RFC3610]的第 5 節和第 10 節。

10.3. ChaCha20 和 Poly1305

ChaCha20 和 Poly1305 組合在一起是 AEAD 模式，在[RFC7539]中定義。這是一種演算法，被定義為非 AES 的密碼，因此不會受到 AES 中未來的任何弱點的影響。這些加密功能旨在快速實現純軟體。

[RFC7539]中定義的 ChaCha20 / Poly1305 AEAD 結構沒有參數化。它需要 256 位元密鑰和 96 位元隨機數，以及明文和附加資料作為輸入，並生成密文作為選項。我們在表 11 中為該演算法定義一個演算法識別符。

Name	Value	Description
ChaCha20/Poly1305	24	ChaCha20/Poly1305 w/ 256-bit key, 128-bit tag

表 11：AES-GCM 演算法的值

密鑰可以從密鑰結構或從接收者結構獲得。實現加密和解密必須驗證密鑰型態，密鑰長度和演算法是否正確並適用於所涉及的實體。

當對此演算法使用 COSE 密鑰時，會進行以下檢查：

- 必須存在'kty'欄位，並且它必須是'對稱'。
- 如果存在'alg'欄位，它必須與正在使用的 ChaCha20 / Poly1305 演算法匹配。
- 如果存在'key_ops'欄位，則在加密時必須包括'encrypt'或'wrap key'。
- 如果存在'key_ops'欄位，則在解密時必須包括'decrypt'或'unwrap key'。

10.3.1. 安全考慮因素

對於演算法的每次調用，密鑰和隨機數值必須是唯一的一對。隨機數計數器被認為是確保它們是唯一的可接受方式。

11. 密鑰衍生函數（KDFs）

KDFs 用於獲取一些隱密值並產生不同的值。隱密值有三種形式：

- 均勻隨機的隱密值：這是由好的隨機數生成器建立的隱密型態。
- 不一致隨機的隱密值：這是由密鑰協定等操作建立的隱密型態。
- 非隨機的隱密值：這是人們生成像密碼之類的隱密型態。

一般的 KDFs 可以很好地處理第一種型態的隱密，可以很好地處理第二種型態的隱密，並且通常對最後一種型態的隱密做得不好。本節中沒有任何 KDFs 旨在處理用於密碼的機密型態。像 PBES2 [RFC8018]這樣的函數需要用於那種型態的隱密。

可以設置相同的 KDF 以不同的方式處理前兩種型態的隱密。第 11.1 節中定義的 KDF 就是這樣一種功能。這反映在為基於 HMAC 的提取和擴充金鑰推衍函數（HKDF）定義的演算法集中。

使用 KDFs 時，包含的一個組件是上下文資訊。上下文資訊用於允許從相同的秘密導出不同的密鑰信息。使用基於上下文的密鑰材料被認為是一種很好的安全實踐。

本文定義單個上下文結構和單個 KDF。這些元素用於本文中定義的所有需要一個 KDF 流程的接收者演算法。這些演算法在第 12.1.2 節，第 12.4.1 節和第 12.5.1 節中定義。

11.1. 基於 HMAC 的提取和擴充密鑰推衍函數（HKDF）

HKDF 密鑰推衍演算法在[RFC5869]中定義。
HKDF演算法採用以下輸入：

隱密值（secret）-- 一個隱密值的共享價值。隱密值可以先前共享，也可以從 Diffie-Hellman（DH）密鑰協定等操作中獲得。

salt -- 用於更改生成過程的可選值。salt 值可以是公共的也可以是私人的。如果 salt 是公共的並且在訊息中攜帶，則使用表 13 中定義的 'salt' 演算法標頭參數。雖然[RFC5869]建議 salt 的長度與底層雜湊值的長度相同，但是任何數量的 salt 都將提高安全性，因為將生成不同的鍵值。此參數通過包含在密鑰計算中而受到保護，無需單獨進行驗證。對於發送的每條訊息，salt 值不必是唯一的。

長度（length）-- 需要生成的輸出位元組數。

上下文資訊（context information）-- 描述將使用結果值的上下文的資訊。使此資訊特定於將要使用材料的上下文，可確保生成的材料始終與該用法相關聯。第 11.2 節中定義的上下文結構由本文中的 KDFs 使用。

PRF -- 在 HKDF 演算法中使用的基礎偽隨機函數。 PRF 被編碼到 HKDF 演算法選擇中。

HKDF 被定義為使用 HMAC 作為基礎 PRF。但是，可以在同一構造中使用其他函數來提供在受約束的世界中更合適的不同 KDF。具體而言，可以使用 AES-CBC-MAC 作為擴充步驟的 PRF，但不能用於提取步驟。當使用正確長度的良好隨機共用密碼時，可以跳過提取步驟。對於 AES 演算法版本，始終跳過提取步驟。

如果隱密值不是均勻隨機的，則不能跳過提取步驟，例如，如果它是 ECDH 密鑰協定步驟的結果。這意味著 AES HKDF 版本不能與 ECDH 一起使用。如果跳過提取步驟，則“salt”值不會用作 HKDF 功能的一部分。

本文中定義的演算法見表 12。

Name	PRF	Description
HKDF SHA-256	HMAC with SHA-256	HKDF using HMAC SHA-256 as the PRF
HKDF SHA-512	HMAC with SHA-512	HKDF using HMAC SHA-512 as the PRF
HKDF AES-MAC-128	AES-CBC-MAC-128	HKDF using AES-MAC as the PRF w/ 128-bit key
HKDF AES-MAC-256	AES-CBC-MAC-256	HKDF using AES-MAC as the PRF w/ 256-bit key

表 12：HKDF 演算法

Name	Label	Type	Algorithm	Description
salt	-20	bstr	direct+HKDF-SHA-256, direct +HKDF-SHA-512, direct+HKDF-AES-128, direct+HKDF-AES-256, ECDH-ES+HKDF-256, ECDH-ES+HKDF-512, ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-ES+A128KW, ECDH-ES+A192KW, ECDH-ES+A256KW, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	Random salt

表 13：HKDF 演算法參數

11.2. 上下文資訊結構

上下文資訊結構用於確保衍生的密鑰材料“綁定”到交易的上下文。此處使用的上下文資訊結構基於[SP800-56A]中定義的結構。通過使用 CBOR 對上下文資訊結構進行編碼，我們自動獲得通過使用 ASN.1 獲得的欄位的相同型態和長度分離。這意味著不需要對基本元素的長度進行編碼，因為它是通過 JOSE 中使用的編碼來完成的（[RFC7518]的第 4.6.2 節）。

上下文資訊結構是指 PartyU 和 PartyV 作為進行密鑰衍生的雙方。除非應用程式協定定義不同，否則我們將 PartyU 分配給正在建立訊息的實體，並將 PartyV 分配給正在接收訊息的實體。通過這種關聯，將為每個方向導出不同的密鑰，因為上下文資訊在每個方向上是不同的。

上下文結構是根據兩個實體都知道的資訊構建的。這些資訊可以從各種來源獲得：

- 可以由應用程式定義欄位。這通常用於為聚會分配固定名稱，但它可以用於其他項目，如隨機數。
- 可以通過使用輸出來定義欄位。這方面的例子是正在生成的演算法和密鑰大小。

- 可以通過訊息中的參數定義欄位。我們在表 14 中定義一組參數，這些參數可用於攜帶與上下文結構相關聯的值。這方面的例子是識別符和隨機數值。這些參數被設計為放置在接收者結構的未受保護的儲存區中；它們不需要位於受保護的儲存區中，因為它們已經包含在上下文結構中，因此已經包含在加密計算中。

Name	Label	Type	Algorithm	Description
PartyU identity	-21	bstr	direct+HKDF-SHA-256, direct+HKDF-SHA-512, direct+HKDF-AES-128, direct+HKDF-AES-256, ECDH-ES+HKDF-256, ECDH-ES+HKDF-512, ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-ES+A128KW, ECDH-ES+A192KW, ECDH-ES+A256KW, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	Party U identity information
PartyU nonce	-22	bstr / int	direct+HKDF-SHA-256, direct+HKDF-SHA-512, direct+HKDF-AES-128, direct+HKDF-AES-256, ECDH-ES+HKDF-256, ECDH-ES+HKDF-512, ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-ES+A128KW, ECDH-ES+A192KW, ECDH-ES+A256KW, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	Party U provided nonce
PartyU other	-23	bstr	direct+HKDF-SHA-256, direct+HKDF-SHA-512, direct+HKDF-AES-128, direct+HKDF-AES-256, ECDH-ES+HKDF-256, ECDH-ES+HKDF-512, ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-ES+A128KW, ECDH-ES+A192KW, ECDH-ES+A256KW, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	Party U other provided information
PartyV identity	-24	bstr	direct+HKDF-SHA-256, direct+HKDF-SHA-512, direct+HKDF-AES-128, direct+HKDF-AES-256, ECDH-ES+HKDF-256, ECDH-ES+HKDF-512, ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-ES+A128KW, ECDH-ES+A192KW, ECDH-ES+A256KW, ECDH-SS+A128KW, ECDH-	Party V identity information

			SS+A192KW, ECDH-SS+A256KW	
PartyV nonce	-25	bstr / int	direct+HKDF-SHA-256, direct+HKDF-SHA-512, direct+HKDF-AES-128, direct+HKDF-AES-256, ECDH-ES+HKDF-256, ECDH-ES+HKDF-512, ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-ES+A128KW, ECDH-ES+A192KW, ECDH-ES+A256KW, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	Party V provided nonce
PartyV other	-26	bstr	direct+HKDF-SHA-256, direct+HKDF-SHA-512, direct+HKDF-AES-128, direct+HKDF-AES-256, ECDH-ES+HKDF-256, ECDH-ES+HKDF-512, ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-ES+A128KW, ECDH-ES+A192KW, ECDH-ES+A256KW, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	Party V other provided information

表 14：上下文演算法參數

我們定義一個 CBOR 物件來保存上下文資訊。該物件稱為 COSE_KDF_Context。該物件基於 CBOR 陣列型態。陣列中的欄位是：

演算法 ID（AlgorithmID）：該欄位表示將使用密鑰材料的演算法。這通常是密鑰包裝演算法識別符或內容加密演算法識別符。值來自“COSE 演算法”註冊表。該欄位必須存在。該欄位存在於上下文資訊中，因此如果相同的環境用於不同的演算法，則將為這些演算法中的每一個生成完全不同的密鑰。這種做法意味著如果演算法 A 被破壞並因此更容易找到，則演算法 B 的導出的密鑰與演算法 A 導出的密鑰不同。

PartyUInfo：該欄位包含有關 partyU 的信息.PartyUInfo 被編碼為 CBOR 陣列。 PartyUInfo 的元素按所示順序編碼。

PartyUInfo 陣列的元素是：

識別符（identity）：包含 party U 的識別符資訊。

識別符可以用兩種方式中其一分配。首先，協定可以根據角色分配識別符。例如，“客戶端”和“伺服器”的角色可以分配給協定中的不同實體。然後，每個實體將為他們發送或接收的資料使用正確的標籤。協定分配標識的第二種方式是使用基於命名系統的名稱（即 DNS，X.509 名稱）。

我們定義一個演算法參數 'PartyU identity'，可用於在訊息中攜帶識別符資訊。然而，識別符資訊通常被稱為協定的一部分，因此可以推斷而不是明確。如果在訊息中攜帶識別符資訊，應用程式應該有一種驗證提供的識別符資訊的方法。識別符資訊不需要指定，在這種情況下設置為零值。

隨機值（nonce）：包含一個隨機值。隨機數可以是協定隱含的，也可以作為未受保護的標頭中的值攜帶。

我們定義一個演算法參數'PartyU nonce'，可以用來在訊息中攜帶這個值；但是，隨機值可以由應用程式確定，並且值可以從其他地方確定。

此選項不需要指定，在這種情況下設置為零值。

其他（other）：包含協定定義的其他信息。此選項不需要指定，在這種情況下設置為零值。

PartyVInfo：該欄位包含有關 party V 的信息。結構的內容與 PartyUInfo 相同，但對於 party V。

SuppPubInfo：該欄位包含雙方都知道的公共資訊。

密鑰資料長度（keyDataLength）：設置為所需輸出值的位元數。這種做法意味著如果演算法 A 可以使用兩個不同的

密鑰長度，則為較長密鑰大小導出的密鑰將不包含用於較短密鑰大小的密鑰作為前綴。

保護（protected）：此欄位包含受保護的參數欄位。如果受保護欄位中沒有元素，則使用零長度 bstr。

其他（other）：此欄位用於應用程式定義的自由格式資料。一個範例是應用程式可以定義兩個不同的字串，以便為資料流與控制流生成不同的密鑰。此欄位是可選的，僅在應用程式為此資訊定義結構時才會出現。定義它的應用程式應該使用 CBOR 對資料進行編碼，以便正確包含型態和長度。

SuppPrivInfo：該欄位包含相互已知的私人信息的私人信息。這種信息的一個例子是預先存在的共用密碼。（例如，這可以與 ECDH 密鑰協定結合使用，以提供識別符的次要證明。）該欄位是可選的，只有在應用程式為此信息定義結構時才會出現該欄位。定義它的應用程式應該使用 CBOR 對資料進行編碼，以便正確包含型態和長度。

以下 CDDL 片段對應於上面的文本。

```
PartyInfo = (
    identity : bstr / nil,
    nonce : bstr / int / nil,
    other : bstr / nil
)

COSE_KDF_Context = [ AlgorithmID : int / tstr,
    PartyUInfo : [ PartyInfo ],
    PartyVInfo : [ PartyInfo ],
    SuppPubInfo : [
        keyDataLength : uint,
        protected : empty_or_serialized_map,
        ? other : bstr
    ],
    ? SuppPrivInfo : bstr
]
```

12. 內容密鑰分配方法

內容密鑰分配方法(接收者演算法)可以定義為許多不同的類別。COSE 能夠支援許多類別的接收者演算法。在本節中，列出許多

類別，然後為每個類別指定一組演算法。此處使用的接收者演算法階級的名稱與[RFC7516]中定義的名稱相同。其他規範對接收方演算法階級使用不同的術語，或者不支援某些接收方演算法階級。

12.1. 直接加密

直接加密類演算法在發送方和接收方之間共用密碼，該隱密值直接或在操作之後用作 CEK。使用直接加密模式時，它必須是訊息上使用的唯一模式。

接收者的 COSE_Recipient 結構組織如下：

- 'protected'欄位必須是零長度項，除非它用於計算內容密鑰。
- 必須存在'alg'參數。
- 應該存在辨識共用密碼的參數。
- “ciphertext”欄位必須是零長度項。
- “recipients”欄位必須不存在。

12.1.1. 直接金鑰

這個接收者演算法是最簡單的；辨識的金鑰直接用作訊息中下一層的金鑰。沒有為此演算法定義演算法參數。演算法識別符值在表 15 中分配。

使用此演算法時，受保護的欄位必須為零長度。鍵值型態必須是“對稱”。

Name	Value	Description
direct	-6	Direct use of CEK

表 15:直接金鑰

12.1.1.1. 安全考慮因素

此接收者演算法有幾個潛在的問題需要考慮：

- 這些金鑰需要有一些方法可以定期更新。本文中指定的所有內容加密演算法都限制密鑰的使用次數，而不會顯著降低安全性。
- 這些密鑰需要專用於單個演算法。當單個密鑰用於多種不同的演算法時，隨著時間的推移已經出現許多攻擊。其中一個例子是對 CBC 加密模式和 CBC-MAC 認證模式使用單個密鑰。
- 中斷一條訊息意味著所有訊息都被破壞。如果攻擊者成功確定單一訊息的密鑰，則還確定所有訊息的密鑰。

12.1.2. 直接金鑰與 KDF

這些接收者演算法在雙方之間採用共同的共用密碼，並應用 HKDF 函數（第 11.1 節），使用第 11.2 節中定義的上下文結構將共用密碼轉換為 CEK。'protected' 欄位可以是非零長度。必須存在 HKDF 的 'salt' 參數或上下文結構的 'PartyU nonce' 參數。salt / nonce 參數可以隨機生成或確定生成。要求它它是所討論的共用密碼的唯一值。

如果隨機生成 salt / nonce 值，則建議隨機值的長度與 HKDF 下的雜湊函數的長度相同。雖然沒有辦法保證它是獨一無二的，但它很有可能是唯一的。如果確定性地生成 salt / nonce 值，則可以保證它是唯一的，因此沒有長度要求。如果使用相同的金鑰，則必須為每條訊息使用新的 IV。可以以可預測的方式，隨機方式或不可預測的方式（即，加密計數器）來修改 IV。

用於密鑰的 IV 也可以從生成密鑰的相同 HKDF 功能生成。如果 HKDF 用於生成 IV，則演算法識別符設置為 “IVGENERATION”。

當使用這些演算法時，密鑰型態必須是“對稱的”。

本文中定義的演算法集可在表 16 中找到。

Name	Value	KDF	Description
direct+HKDF-SHA-256	-10	HKDF SHA-256	Shared secret w/ HKDF and SHA-256
direct+HKDF-SHA-512	-11	HKDF SHA-512	Shared secret w/ HKDF and SHA-512
direct+HKDF-AES-128	-12	HKDF AES-MAC-128	Shared secret w/ AES-MAC 128-bit key
direct+HKDF-AES-256	-13	HKDF AES-MAC-256	Shared secret w/ AES-MAC 256-bit key

表 16：用 KDF 的直接金鑰

對此演算法使用 COSE 密鑰時，將進行以下檢查：

- 必須存在'kty'欄位，它必須是'對稱'。
- 如果存在'alg'欄位，它必須與正在使用的演算法匹配。
- 如果'key_ops'欄位存在，它必須包含'deriveKey'或'deriveBits'。

12.1.2.1. 安全考慮因素

共用密碼需要有一些方法隨著時間的推移定期更新。共用密碼構成信任的基礎。雖然沒有直接使用，但仍應按計劃輪調。雖然這些方法不提供完美的正向密碼，因為對於生成的所有密鑰使用相同的共用密碼，如果發現任何單個訊息的密鑰，則僅使用該衍生密鑰的訊息（或一系列訊息）受到損害。新的密鑰衍生步驟將生成一個新密鑰，該密鑰需要相同的工作量才能獲得密鑰。

12.2. 密鑰包裝

在密鑰包裝模式中，CEK 是隨機生成的，然後該密鑰由發送者和接收者之間的共用密碼加密。所有當前定義的 COSE 密鑰包裝演算法都是 AE 演算法。如果系統具有進行隨機密鑰生成的任何能力，則密鑰包裝模式被認為優於直接加密。這是因為共用密碼用於包裝隨機資料而不是具有某種程度組織的資料，並且實際上可能重複相同的內容。密鑰包裝的使用失去了由直接加密演算法提供的弱資料源。

接收者的 COSE_Encrypt 結構組織如下：

- 如果密鑰包裝演算法是 AE 演算法，則“protected”欄位必須不存在。
- “recipients”欄位通常不存在，但可以使用。應用程式必須處理存在的接收者欄位，而不能解密該接收者是一種可接受的處理方式。未能處理訊息是不可接受的處理方式。
- 要加密的明文是下一層（通常是內容層）的關鍵。
- 至少“unprotected”欄位必須包含“alg”參數，並且應該包含辨識共用密碼的參數。

12.2.1. AES 密鑰包裝

AES 密鑰包裝演算法在[RFC3394]中定義。該演算法使用 AES 密鑰來包裝 64 位元的倍數值。因此，它可以用於包裝本文中定義的任何內容加密演算法的密鑰。該演算法需要一個固定參數，即初始值。這固定為[RFC3394]第 2.2.3.1 節中規定的值。沒有公共參數因每次調用而異。受保護的標頭欄位必須為空。

密鑰可以從密鑰結構或從接收者結構獲得。實現加密和解密必須驗證密鑰型態，密鑰長度和演算法是否正確並適用於所涉及的實體。

對此演算法使用 COSE 密鑰時，將進行以下檢查：

- 必須存在'kty'欄位，它必須是'對稱'。
- 如果存在'alg'欄位，它必須與正在使用的 AES Key Wrap 演算法匹配。
- 如果存在'key_ops'欄位，則在加密時必須包括'encrypt'或'wrap key'。
- 如果存在'key_ops'欄位，則在解密時必須包括'decrypt'或'unwrap key'。

Name	Value	Key Size	Description
A128KW	-3	128	AES Key Wrap w/ 128-bit key
A192KW	-4	192	AES Key Wrap w/ 192-bit key
A256KW	-5	256	AES Key Wrap w/ 256-bit key

表 17：AES 密鑰包裝演算法的值

12.2.1.1. AES-KW 的安全注意事項

共用密碼需要有一些方法隨著時間的推移定期更新。共用密碼是信任的基礎。

12.3. 密鑰傳輸

密鑰傳輸模式在某些標準中也稱為密鑰加密模式。密鑰傳輸模式與密鑰包裝模式的不同之處在於它使用非對稱加密演算法而不是對稱加密演算法來保護密鑰。本文未定義任何密鑰傳輸模式演算法。

使用密鑰傳輸演算法時，接收者的 COSE_Encrypt 結構組織如下：

- “protected”欄位必須不存在。
- 要加密的明文是下一層（通常是內容層）的密鑰。
- 至少“unprotected”欄位必須包含’alg’參數，並且應該包含辨識非對稱密鑰的參數。

12.4. 直接金鑰協定

“直接金鑰協定”類別的接收方演算法使用金鑰協商方法來建立共用密碼。然後將 KDF 應用於共用密碼以導出用於保護資料的金鑰。此金鑰通常用作 CEK 或 MAC 金鑰，但如果使用兩個以上的層，則可用於其他目的（參見附錄 B）。

最常用的密鑰協定演算法是 Diffie-Hellman，但存在其他變體。由於 COSE 是專為存儲和轉送環境而非在線環境而設計的，因此許多 DH 變體不能用作接收者的訊息，因此無法提供任何動態密鑰

材料。這樣做的一個副作用是完美的前向保密（見[RFC4949]）是不可能實現的。靜態密鑰將始終用於 COSE 物件的接收者。

支援的兩種 DH 變體是：

短暫靜態（ES）DH：訊息的發送者建立一次性 DH 密鑰並為接收者使用靜態密鑰。短暫發送者密鑰的使用意味著不需要額外的隨機輸入，因為這是為每條訊息隨機生成的。

靜態 – 靜態 DH：靜態密鑰用於發送者和接收者。靜態密鑰的使用允許接收者獲得訊息的弱版本的資料發起。當使用靜態 - 靜態密鑰協定時，則需要 KDF 的一些唯一資料以確保為每個訊息建立不同的密鑰。

使用直接金鑰協商模式時，訊息中必須只有一個接收者。此方法直接建立密鑰，這使得難以與其他接收者混合。如果需要多個接收者，則需要使用帶密鑰包裝的版本。

接收者的 COSE_Encrypt 結構組織如下：

- 標題必須至少包含'alg'參數，並且應該包含一個辨識接收者非對稱密鑰的參數。
- 標題應該辨識靜態 - 靜態版本的發送者密鑰，並且必須包含短暫靜態版本的發送者短暫密鑰。

12.4.1. ECDH

ECDH 的數學可以在[RFC6090]中找到。在本文中，演算法擴充為與[RFC7748]中定義的兩條曲線一起使用。

ECDH 通過以下參數化：

- 曲線類型/曲線：所選曲線不僅控制共用密碼的大小，還控制計算共用密碼的數學。選擇的曲線還控制曲線中的點如何表示以及曲線上的辨識點會發生什麼。在此說明書中，我們允許使用許多不同的曲線。表 22 中定義一組曲線。用於獲得計算機密的數學基於所選曲線而不是 ECDH 演算法。因此，不需要為每條曲線定義新演算法。

- 共用密碼的計算隱密值：一旦知道計算的隱密值，就需要將結果值轉換為位元組字串以運行 KDF。 x 坐標用於本文中定義的所有曲線。對於曲線 X25519 和 X448，直接使用結果值，因為它是已知長度的位元組串。對於 P-256，P-384 和 P-521 曲線，x 坐標通過[RFC8017]中定義的 I2OSP 函數運行，使用與第 8.1 節中定義的 n 相同的計算。
- 短暫靜態或靜態 – 靜態：密鑰協定過程可以使用發送方的靜態密鑰或臨時密鑰完成。使用臨時密鑰時，發送方必須為每個密鑰協定操作生成一個新的臨時密鑰。臨時密鑰放在“臨時密鑰”參數中，並且必須存在於使用臨時密鑰的所有演算法識別符。使用靜態密鑰時，發送方必須生成新的隨機值或建立唯一值。對於使用的 KDF，這意味著 HKDF 的“salt”參數（表 13）或上下文結構的“PartyU 隨機數”參數（表 14）必須存在（如果需要，兩者都可以存在）。對於正在使用的密鑰對，參數中的值必須是唯一的。可以使用為每個靜態 - 靜態操作遞增的全局計數器並使用結果值。使用靜態密鑰時，應該向接收者辨識靜態密鑰。可以通過提供密鑰（“靜態密鑰”）或通過提供靜態密鑰的密鑰識別符（“靜態密鑰 id”）來辨識靜態密鑰。這兩個參數都在表 19 中定義。
- 密鑰推衍演算法：ECDH 密鑰協定過程的結果不提供統一的隨機隱密值。因此，它需要通過 KDF 運行才能生成可用的密鑰。通過 KDF 處理隱密值還允許引入上下文材料：密鑰將如何使用以及靜態密鑰協定的一次性材料。本文中定義的所有演算法都使用第 11.1 節中定義的 HKDF 演算法之一，以及第 11.2 節中定義的上下文結構。
- 密鑰包裝演算法：不使用密鑰包裝演算法。這在表 18 中表示為“無”。上下文結構的密鑰大小是內容層加密演算法大小。

本文中定義的一組直接 ECDH 演算法見表 18。

Name	Value	KDF	Ephemeral-Static	Key Wrap	Description
ECDH-ES + HKDF-256	-25	HKDF - SHA-256	yes	none	ECDH ES w/ HKDF - generate key directly
ECDH-ES + HKDF-512	-26	HKDF - SHA-512	yes	none	ECDH ES w/ HKDF - generate key directly
ECDH-SS + HKDF-256	-27	HKDF - SHA-256	no	none	ECDH SS w/ HKDF - generate key directly
ECDH-SS + HKDF-512	-28	HKDF - SHA-512	no	none	ECDH SS w/ HKDF - generate key directly

表 18：ECDH 演算法的值

Name	Label	Type	Algorithm	Description
ephemeral key	-1	COSE_Key	ECDH-ES+HKDF-256, ECDH-ES+HKDF-512, ECDH-ES+A128KW, ECDH-ES+A192KW, ECDH-ES+A256KW	Ephemeral public key for the sender
static key	-2	COSE_Key	ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	Static public key for the sender
static key id	-3	bstr	ECDH-SS+HKDF-256, ECDH-SS+HKDF-512, ECDH-SS+A128KW, ECDH-SS+A192KW, ECDH-SS+A256KW	Static public key identifier for the sender

表 19：ECDH 演算法參數

該文定義與曲線 P-256，P-384，P-521，X25519 和 X448 一起使用的這些演算法。實現必須驗證密鑰型態和曲線是否正確。不同的曲線僅限於不同的密鑰型態。實現必須驗證曲線和演算法是否適合所涉及的實體。

對此演算法使用 COSE 密鑰時，將進行以下檢查：

- 必須存在'kty'欄位，它必須是'EC2'或'OKP'。
- 如果存在'alg'欄位，它必須與正在使用的密鑰協定演算法匹配。
- 如果存在'key_ops'欄位，它必須包括私鑰的'衍生密鑰'或'衍生位元'。
- 如果存在'key_ops'欄位，則公鑰必須為空。

12.4.2. 安全考慮因素

有一種方法可以檢查外部實體提供的點是否有效。對於'EC2'鍵值格式，可以通過檢查 x 和 y 值是否形成曲線上的點來完成。對於'OKP'格式，沒有簡單的方法來進行點驗證。

考慮要求將兩個實體的公鑰作為密鑰推衍過程的一部分提供（如[RFC7748]第 6.1 節中所建議的那樣）。沒有這樣做是因為 COSE 用於儲存和轉發格式而不是線上密鑰交換。為了使這成為一個問題，必須惡意選擇接收者公鑰或者發送者必須是惡意的。在任何一種情況下，無論如何所有安全都會發散。

當密鑰從不信任移動到受信任時（由最終用戶或負責在密鑰上建立信任語句的實體），建議擁有與公鑰關聯的私鑰的證明。

12.5. 密鑰包裝的密鑰協定

密鑰包裝的密鑰協定使用隨機生成的 CEK。然後使用密鑰包裝演算法和從密鑰協定演算法計算的共用密碼導出的密鑰對 CEK 進行加密。這個功能是：

接收者的 COSE_Encrypt 結構組織如下：

- “protected”欄位被輸入 KDF 上下文結構。
- 要加密的明文是下一層（通常是內容層）的密鑰。
- 'alg'參數必須出現在圖層中。
- 應該存在辨識接收者密鑰的參數。應該存在辨識發送者密鑰的參數。

12.5.1. ECDH

這些演算法在表 20 中定義。

具有密鑰協定的 ECDH 通過與 ECDH 相同的參數進行參數化;請參見第 12.4.1 節，並進行以下修改：

- 密鑰包裝演算法：支援第 12.2.1 節中定義的任何密鑰包裝演算法。用於密鑰包裝演算法的密鑰的大小被送到 KDF。表 20 中列出一組識別符。

Name	Value	KDF	Ephemeral-Static	Key Wrap	Description
ECDH-ES + A128KW	-29	HKDF - SHA-256	yes	A128KW	ECDH ES w/ Concat KDF and AES Key Wrap w/ 128-bit key
ECDH-ES + A192KW	-30	HKDF - SHA-256	yes	A192KW	ECDH ES w/ Concat KDF and AES Key Wrap w/ 192-bit key
ECDH-ES + A256KW	-31	HKDF - SHA-256	yes	A256KW	ECDH ES w/ Concat KDF and AES Key Wrap w/ 256-bit key
ECDH-SS + A128KW	-32	HKDF - SHA-256	no	A128KW	ECDH SS w/ Concat KDF and AES Key Wrap w/ 128-bit key
ECDH-SS + A192KW	-33	HKDF - SHA-256	no	A192KW	ECDH SS w/ Concat KDF and AES Key Wrap w/ 192-bit key
ECDH-SS + A256KW	-34	HKDF - SHA-256	no	A256KW	ECDH SS w/ Concat KDF and AES Key Wrap w/ 256-bit key

表 20：使用密鑰包裝 ECD 演算法的值

對此演算法使用 COSE 密鑰時，將進行以下檢查：

- 必須存在'kty'欄位，它必須是'EC2'或'OKP'。
- 如果存在'alg'欄位，它必須與正在使用的密鑰協定演算法匹配。
- 如果存在'key_ops'欄位，它必須包括私鑰的'衍生密鑰'或'衍生位元'。
- 如果存在'key_ops'欄位，則公鑰必須為空。

13. 密鑰物件參數

COSE_Key 物件定義一種保持單一密鑰物件的方法。仍然需要定義各個密鑰型態的成員。本文的這一部分是我們為特定密鑰型態定義初始成員集的地方。

對於每種密鑰型態，我們定義公共和私有成員。公共成員是傳遞給他人使用的內容。私有成員允許個人歸檔密鑰。但是，在某些情況下，私鑰可以分配給協定中的實體。範例包含：具有差的亂數生成的實體，用於多播類型操作的集中型密鑰建立，以及其中共用密碼被用於授權目的的持有人訊標的協定。

密鑰型態由 COSE_Key 物件的“kty”成員辨識。在本文中，我們為成員定義四個值：

Name	Value	Description
OKP	1	Octet Key Pair
EC2	2	Elliptic Curve Keys w/ x- and y-coordinate pair
Symmetric	4	Symmetric Keys
Reserved	0	This value is reserved

表 21：密鑰型態的值

13.1. 橢圓曲線密鑰

為橢圓曲線密鑰定義兩個不同的密鑰結構。一個版本使用 x 坐標和 y 坐標，可能使用點壓縮（'EC2'）。這是[RFC5480]中使用的傳統 EC 點表示。另一個版本僅使用 x 坐標，因為 y 坐標不是重新計算，就是不需要進行密鑰協定操作（'OKP'）。

應用程式必須檢查曲線和密鑰型態是否一致，如果不是，則拒絕密鑰。

Name	Value	Key Type	Description
P-256	1	EC2	NIST P-256 also known as secp256r1
P-384	2	EC2	NIST P-384 also known as secp384r1
P-521	3	EC2	NIST P-521 also known as secp521r1
X25519	4	OKP	X25519 for use w/ ECDH only
X448	5	OKP	X448 for use w/ ECDH only
Ed25519	6	OKP	Ed25519 for use w/ EdDSA only
Ed448	7	OKP	Ed448 for use w/ EdDSA only

表 22：橢圓曲線

13.1.1. 雙坐標曲線

發送 ECs 的傳統方式是發送 x 坐標和 y 坐標或 x 坐標以及 y 坐標的符號位元。由於潛在的 IPR 問題，IETF 中未推薦後一種的編碼。但是，對於受約束環境中的操作，通過不發送 y 坐標來收縮訊息的能力可能是有用的。

對於具有兩個坐標的 EC 密鑰，'kty'成員設置為 2（EC2）。表 23 中總結本節中定義的密鑰參數。為此密鑰型態定義的成員是：

crv：包含要與密鑰一起使用的曲線的識別符。本文中為此今鑰型態定義的曲線可在表 22 中找到。其他曲線可能會在將來註冊，也可以使用私有曲線。

x：這包含 EC 點的 x 坐標。整數轉換為[SEC1]中定義的八位元字串。必須保留前置字元零的八位元組。

y：它包含正負號位元或 EC 點的 y 坐標值。對值 y 進行編碼時，整數將轉換為八位元字串（如[SEC1]中所定義）並編碼為 CBOR bstr。必須保留前置字元零的八位元組。也支援壓縮點編碼。計算正負號位元，如[SEC1]的橢圓曲線點到八位元字串轉換函數所示。如果正負號位元為零，則將 y 編碼為 CBOR 假值；否則，將 y 編碼為 CBOR 真值。不支援無限遠點的編碼。

d：包含私鑰。

對於公鑰，必須在結構中存在“crv”，“x”和“y”。對於私鑰，要求“crv”和“d”存在於結構中。對於私鑰，建議“x”和“y”

也存在，但可以從所需元素重新計算它們，省略它們可以節省空間。

Key Type	Name	Label	CBOR Type	Description
2	crv	-1	int / tstr	EC identifier - Taken from the "COSE Elliptic Curves" registry
2	x	-2	bstr	x-coordinate
2	y	-3	bstr / bool	y-coordinate
2	d	-4	bstr	Private key

表 23：EC 密鑰參數

13.2. 八位元組密鑰對

為八位元組密鑰對（OKP）定義新的密鑰型態。不要假設使用此型態的密鑰是橢圓曲線。此密鑰型態可用於其他曲線型態（例如，基於超橢圓曲面的數學）。

表 24 中總結本節中定義的關鍵參數。為此密鑰型態定義的成員是：

crv：包含要與密鑰一起使用的曲線的識別符。本文中為此密鑰型態定義的曲線可以在表 22 中找到。其他曲線可以在將來註冊，也可以使用私有曲線。

x：這包含 EC 點的 x 坐標。八位元字串表示 x 的小端讀取型編碼。

d：包含私鑰。

對於公鑰，要求“crv”和“x”存在於結構中。對於私鑰，要求“crv”和“d”存在於結構中。對於私鑰，建議“x”也存在，但可以從所需元素重新計算，省略它可以節省空間。

Name	Key Type	Label	Type	Description
crv	1	-1	int / tstr	EC identifier - Taken from the "COSE Key Common Parameters" registry
x	1	-2	bstr	x-coordinate
d	1	-4	bstr	Private key

表 24：八位元組密鑰對參數

13.3. 對稱密鑰

有時，需要在實體之間傳輸對稱密鑰。這種密鑰結構允許這種情況發生。

對於對稱密鑰，'kty'成員設置為 4（對稱）。為此密鑰型態定義的成員是：

k：這包含密鑰的值。

此密鑰結構沒有僅包含公共成員的表單。由於預計這種密鑰結構將被傳輸，因此必須注意它不會被意外或不安全地傳輸。對於對稱密鑰，需要在結構中存在'k'。

Name	Key Type	Label	Type	Description
k	4	-1	bstr	Key Value

表 25：對稱密鑰參數

14. CBOR 編碼器限制

文件需要對CBOR編碼器如何工作施加的限制已經嘗試限制位置的數量。我們設法將其縮小到以下限制：

- 該限制適用於 Sig_structure，Enc_structure 和 MAC_structure 的編碼。

- “Canonical CBOR”（RFC 7049 第 3.9 節）的規則必須在這些地方使用。需要強制執行的主要規則是，必須對這些結構中的所有長度進行編碼，使得它們使用確定的長度，並使用最小長度編碼。
- 應用程式不得生成具有相同標籤的訊息，該標籤在單個映射中使用兩次作為鍵值。應用程式不得解析和處理具有相同標籤的訊息，該標籤在單個映射中使用兩次作為鍵值。應用程式可以通過使用將解析步驟失敗的解析器或使用將所有鍵值傳遞給應用程式的解析器來強制執行解析和處理要求，並且應用程式可以執行對重複鍵值的檢查。

15. 應用程式分析注意事項

本文旨在提供一組安全服務，但不提供特定用法的實現要求。提供互操作性要求，以說明如何使用每個單獨的服務以及如何將演算法用於互操作性。關於需要哪些演算法和哪些服務的要求緩發到每個應用程式。
可以在[OSCOAP]中找到配置文件的範例，其中正在開發兩個配置文件。一個用於自己攜帶內容，另一個用於攜帶與 CoAP 標題組合的內容。

旨在建立本文的概要文件，以定義該特定應用程式的互操作性要求。本節提供一組在分析本文時需要考慮的準則和主題。

- 應用程式需要確定它們將使用的本文中定義的訊息集。訊息集直接對應於所需的安全服務集和所需的安全級別。
- 應用程式可以為特定目的定義新的標頭參數。應用程式通常會選擇要使用或不使用的特定標頭參數。例如，應用程式通常會聲明使用 IV 或部分 IV 參數的首選項。如果指定 Partial IV 參數，則應用程式還需要定義如何確定 IV 的固定部分。
- 當應用程式使用外部定義的經過驗證的資料時，他們需要定義資料的編碼方式。本文假設資料將作為位元組流提供。更多信息可以在第 4.3 節中找到。
- 應用程式需要確定要使用的安全演算法集。當選擇要用作強制執行集的演算法時，應考慮在為特定目的選擇兩個演算法時選擇不同類型的演算法。一個例子是選擇 HMACSHA512

和 AES-CMAC 作為不同的 MAC 演算法；這兩種演算法之間的結構差別很大。這意味著一種演算法的弱化不太可能導致其他演算法的弱化。當然，這些演算法不提供相同級別的安全性，因此可能無法與所需的安全功能進行比較。

- 如果允許多種演算法或訊息結構，應用程式可能需要提供某種類型的協商或發現方法。該方法可以簡單到需要預先配置該組演算法以提供內置於協定中的發現方法。S / MIME 提供許多不同的方法來解決應用程式可能遵循的問題：

  * 在訊息中做廣告（S / MIME 功能）[RFC5751]。
  * 憑證中的廣告（功能擴充）[RFC4262]。
  * S / MIME 的最低要求，隨著時間的推移已經更新 [RFC2633] [RFC5751]（請注意[RFC2633]已被[RFC5751]淘汰）。

16. IANA 注意事項

16.1. CBOR 標籤分配

IANA 已從“CBOR 標籤”註冊表中分配以下標籤。COSE_Sign1，COSE_Encrypt0 和 COSE_Mac0 的標籤分配在 1 到 23 的值範圍內（編碼時長度為一個位元組）。 COSE_Sign，COSE_Encrypt 和 COSE_Mac 的標籤分配在 24 到 255 的值範圍內(編碼時長度為兩個位元組）。

分配的標籤見表 1。

16.2. COSE 標題參數註冊表

IANA 建立一個名為“COSE 標頭參數”的新註冊表。已建立註冊表以使用“需要專家審核”註冊過程[RFC8126]。第 16.11 節提供專家指南。應該注意的是，除了專家評論之外，註冊表的某些部分還需要提供規範，可能還有標準 RFC。

註冊表的列是：

名稱：名稱存在，以便更容易參考和討論註冊條目。該值未在協定中使用。名稱在表格中是唯一的。

標籤：這是用於標籤的值。標籤可以是整數或字串。表中的註冊基於所請求標籤的值。1 到 255 之間的整數值和長度為 1 的字串被指定為“標準操作”。從 256 到 65535 的整數值和長度為 2 的字串被指定為“規範要求”。大於 65535 的整數值和長度大於 2 的字串被指定為“專家評論”。-1 到-65536 範圍內的整數值“委託給 COSE 標頭演算法參數註冊表”。小於-65536 的整數值標籤為私有用途。

值型態：它包含標籤值部分的 CBOR 型態。

值註冊表：它包含一個指向註冊表的指標，該註冊表用於包含集合受限的值。

描述：這包含標題欄位的簡要描述。

參考：它包含指向定義標題欄位（公共）的規範的指標。

註冊表的初始內容可以在表 2 和表 27 中找到。此註冊表的“參考”列中的所有條目都指向此文件。
此外，標籤0將標籤為“Reserved”。

16.3. COSE 標頭演算法參數註冊表

IANA 建立一個名為“COSE 標頭演算法參數”的新註冊表。註冊表使用“需要專家審核”註冊程序。專家審核指南見第 16.11 節。

註冊表的列是：

名稱：名稱存在，以便更容易參考和討論註冊條目。該值未在協定中使用。

演算法：此註冊表項用於的演算法。該值取自“COSE 演算法”註冊表。可以在此條目中指定多個演算法。對於表格，演算法/標籤對必須是唯一的。

標籤：這是用於標籤的值。標籤是-1 到-65536 範圍內的整數。

型態：它包含標籤值部分的 CBOR 型態。
描述：這包含標題欄位的簡要描述。

參考：它包含指向定義標題欄位（公共）的規範的指標。

註冊表的初始內容可以在表 13,14 和 19 中找到。此註冊表的“參考”列中的所有條目都指向此文件。

16.4. COSE 演算法註冊表

IANA 建立一個名為“COSE 演算法”的新註冊表。已建立註冊表以使用“需要專家審核”註冊過程。第 16.11 節提供專家指南。應該注意的是，除了專家評審之外，註冊表的某些部分還需要提供規範，可能還有標準跟蹤 RFC。

註冊表的列是：

名稱：可用於辨識文件中的演算法以便於理解的值。這個名字應該是獨一無二的。但是，“值”欄位用於辨識演算法，而不是“名稱”欄位。

值：用於辨識此演算法的值。演算法值必須是唯一的。該值可以是正整數，負整數或字串。 -256 和 255 之間的整數值和長度為 1 的字串被指定為“標準操作”。從-65536 到 65535 的整數值和長度為 2 的字串被指定為“規範要求”。大於 65535 的整數值和長度大於 2 的字串被指定為“專家評論”。小於 -65536 的整數值標籤為私有用途。

描述：演算法的簡短描述。

參考：定義演算法的文件（如果公開可用）。

推薦：IETF 是否有共識建議使用該演算法？合法值為“是”，“否”和“不適用”。

註冊表的初始內容可以在表 5,6,7,8,9,10,11,15,16,17,18 和 20 中找到。此註冊表點“參考”列中的所有條目到這個文件。“推薦”列中的所有條目都設置為“是”。

此外，標籤 0 將標籤為“Reserved”。

注意：本文中的演算法識別符分配已完成，因此正數用於第一層物件（COSE_Sign，COSE_Sign1，COSE_Encrypt，COSE_Encrypt0，COSE_Mac 和 COSE_Mac0）。負數用於第二層物件（COSE_Signature 和 COSE_recipient）。專家評審員應該考慮這種做法，但預計不會受到此先例的限制。

16.5. COSE 密鑰公共參數註冊表

IANA 建立一個名為“COSE 密鑰公共參數”的新註冊表。已建立註冊表以使用“需要專家審核”註冊過程。第 16.11 節提供專家指南。應該注意的是，除了專家評審之外，註冊表的某些部分還需要提供規範，可能還有標準跟蹤 RFC。

註冊表的列是：

名稱：這是一個描述性名稱，可以更容易地引用該項目。它不用於編碼。

標籤：用於辨識此演算法的值。鍵值映射標籤必須是唯一的。標籤可以是正整數，負整數或字串。 0 到 255 之間的整數值和長度為 1 的字串被指定為“標準操作”。從 256 到 65535 的整數值和長度為 2 的字串被指定為“規範要求”。大於 65535 的整數值和長度大於 2 的字串被指定為“專家評論”。
-65536 到-1 範圍內的整數值“用於特定於委託給 COSE 密鑰型態參數註冊表的單個演算法的鍵值參數”。小於-65536 的整數值標籤為私有用途。

CBOR 型態：該欄位包含該欄位的 CBOR 型態。

值註冊表：此欄位表示值來自的註冊表（如果存在）。

描述：該欄位包含該欄位的簡要說明。

參考：它包含指向欄位的公共規範的指標（如果存在）。

此註冊表最初由表 3 中的值填充。此註冊表的“引用”列中的所有條目都指向此文件。

16.6. COSE 鍵值型態參數註冊表

IANA 建立一個名為“COSE 密鑰型態參數”的新註冊表。已建立註冊表以使用“需要專家審核”註冊過程。專家審核指南見第 16.11 節。

該表的列是：

密鑰型態：此欄位包含密鑰型態的描述性字串。

這應該是“COSE 鍵值常見參數”註冊表中的值，並且放在 COSE Key 結構的“kty”欄位中。

名稱：這是一個描述性名稱，可以更容易地引用該項目。它不用於編碼。

標籤：標籤對於鍵值型態的每個值都是唯一的。值的範圍是-65536 到-1。標籤預計將重複用於不同的密鑰。

CBOR 型態：該欄位包含該欄位的 CBOR 型態。

描述：該欄位包含該欄位的簡要說明。

參考：它包含指向欄位的公共規範的指標（如果存在）。

此註冊表最初由表 23,24 和 25 中的值填充。此註冊表的“References”列中的所有條目都指向此文件。

16.7. COSE 密鑰型態註冊表

IANA 建立一個名為“COSE 密鑰型態”的新註冊表。已建立註冊表以使用“需要專家審核”註冊過程。專家審核指南見第 16.11 節。

該表的列是：

名稱：這是一個描述性名稱，可以更容易地引用該項目。名稱必須是唯一的。它不用於編碼。

值：這是用於辨識曲線的值。這些值必須是唯一的。該值可以是正整數，負整數或字串。

描述：該欄位包含曲線的簡要描述。

參考：它包含指向曲線公共規範的指標（如果存在）。

此註冊表最初由表 21 中的值填充。所有這些條目的規範列於此文件。

16.8. COSE 橢圓曲線註冊表

IANA 建立一個名為“COSE 橢圓曲線”的新註冊表。已建立註冊表以使用“需要專家審核”註冊過程。第 16.11 節提供專家指南。應該注意的是，除了專家評審之外，註冊表的某些部分還需要提供規範，可能還有標準跟蹤 RFC。

該表的列是：

名稱：這是一個描述性名稱，可以更容易地引用該項目。它不用於編碼。

值：這是用於標識曲線的值。這些值必須是唯一的。從-256 到 255 的整數值被指定為“標準操作”。從 256 到 65535 和-65536

到-257 的整數值被指定為“規範要求”。超過 65535 的整數值被指定為“專家評審”。小於-65536 的整數值標籤為私有用途。

鍵值型態：指定可與此曲線一起使用的鍵值型態。

描述：該欄位包含曲線的簡要描述。

參考：它包含指向曲線公共規範的指標（如果存在）。

推薦：IETF 是否有共識建議使用該演算法？合法值為“是”，“否”和“不適用”。

此註冊表最初由表 22 中的值填充。此註冊表的“References”列中的所有條目都指向此文件。“Recommended”列中的所有條目都設置為“是”。

16.9. 媒體類型註冊

16.9.1. COSE 安全信息

此部分在“媒體類型”註冊表中註冊“application / cose”媒體類型。這些媒體類型用於指示內容是 COSE 訊息。

型態名稱：應用程式
子型態名稱：cose
所需參數：N / A.
可選參數：cose-type
編碼注意事項：二進制
安全注意事項：請參閱 RFC 8152 的安全注意事項部分。
互操作性考慮因素：N / A.
發布規範：RFC 8152
使用此媒體類型的應用程式：IoT 應用程式通過 HTTP（S）傳輸發送安全內容。
片段識別符注意事項：N / A.
附加資訊：

*此型態的不適用的別名：N / A.
*幻數：N / A.
*檔案延伸：cbor
* 麥金塔文件類型編碼：N / A.
聯繫人和電子郵件地址以獲取更多資訊：
iesg@ietf.org
預期用途：COMMON
使用限制：N / A.
作者：Jim Schaad，ietf @ augustcellars.com
變更控制器：IESG
臨時註冊？沒有

16.9.2. COSE 鍵值媒體類型

本節在“媒體類型”註冊表中註冊“application / cose-key”和“application / cose-key-set”媒體類型。這些媒體類型分別用於指示內容是 COSE_Key 或 COSE_KeySet 物件。

註冊'application / cose-key'的模板是：
型態名稱：應用程式
子型態名稱：cose-key
所需參數：N / A.
可選參數：N / A.
編碼注意事項：二進制
安全注意事項：請參閱安全注意事項部分 RFC 8152。
互操作性考慮因素：N / A.
發布規範：RFC 8152
使用此媒體類型的應用程式：基於 COSE 的分配物聯網應用的關鍵。
片段識別符注意事項：N / A.
附加資訊：
*此類型的不適用的別名：N / A.
*幻數：N / A.
*檔案延伸：cbor
* 麥金塔文件類型編碼：N / A.
聯繫人和電子郵件地址以獲取更多資訊：iesg@ietf.org

預期用途：COMMON
使用限制：N / A.
作者：Jim Schaad，ietf @ augustcellars.com
變更控制器：IESG
臨時註冊？沒有

註冊'application / cose-key-set'的模板是：
型態名稱：應用程式
子型態名稱：cose-key-set
所需參數：N / A.
可選參數：N / A.
編碼注意事項：二進制
安全注意事項：請參閱 RFC 8152 的安全注意事項部分。
互操作性考慮因素：N / A.
發布規範：RFC 8152
使用此媒體類型的應用程式：為物聯網應用程式分配基於 COSE 的密鑰。
片段識別符注意事項：N / A.
附加資訊：
*此類型的不適用的別名：N / A.
*幻數：N / A.
*檔案延伸：cbor
* 麥金塔文件類型編碼：N / A.
聯繫人和電子郵件地址以獲取更多資訊：
iesg@ietf.org
預期用途：COMMON
使用限制：N / A.
作者：Jim Schaad，ietf @ augustcellars.com
變更控制器：IESG
臨時註冊？沒有

16.10.CoAP 內容格式註冊表

IANA 已將以下條目添加到“CoAP Content-Formats”註冊表中。

Media Type	Encoding	ID	Reference
application/cose; cose-type="cose-sign"		98	[RFC8152]
application/cose; cose-type="cose-sign1"		18	[RFC8152]
application/cose; cose-type="cose-encrypt"		96	[RFC8152]
application/cose; cose-type="cose-encrypt0"		16	[RFC8152]
application/cose; cose-type="cose-mac"		97	[RFC8152]
application/cose; cose-type="cose-mac0"		17	[RFC8152]
application/cose-key		101	[RFC8152]
application/cose-key-set		102	[RFC8152]

表 26：COSE 的 CoAP 內容格式

16.11. 專家評審說明

本文中建立的所有 IANA 註冊管理機構均被定義為專家審核。本節提供一些關於專家應該尋找什麼的一般指導原則，但是由於某種原因它們被指定為專家，所以它們應該被賦予相當大的自由度

專家評審員應考慮以下幾點：

- 不鼓勵蹲點。鼓勵審閱者獲取足夠的註冊請求信息，以確保用法不會復制已註冊的用戶，並且該點很可能在部署中使用。標籤為私人使用的區域用於測試目的和封閉環境；不應將其他範圍內的編碼點分配用於測試。
- 標準跟蹤範圍的點分配需要規範。規範應存在規範要求的範圍，但在規範可用之前的早期分配被認為是允許的。如果要求以可互操作的方式在封閉環境之外使用，則需要先到先服務範圍的規格。如果未提供規範，則所提供的描述需要具有足夠的信息來辨識該點的用途。
- 專家在批准點分配時應考慮到欄位的預期用途。標準跟蹤文件的範圍並不意味著標準跟蹤文件不能在該範圍之外分配點。編碼值的長度應該與該長度剩餘的碼點數量，它將使用的設備大小以及編碼為該大小的剩餘碼點數量進行權衡。

- 當註冊演算法時，不鼓勵進行無意義註冊。一種方法是要求註冊提供有關演算法安全性分析的其他文件。應該考慮的另一件事是從加密論壇研究組（CFRG）請求對演算法的意見。不應註冊不符合社群和訊息結構的安全要求的演算法。

17. 安全考慮因素

本規範的實施者需要考慮許多安全因素。特定於單一演算法的安全性考慮因素放在演算法描述的旁邊。雖然這裡強調一些注意事項，但可以在參考文獻中列出的文件中找到其他注意事項。

實現需要保護任何個人的私鑰材料。本文中的某些情況需要在此問題上突出顯示。

- 對兩種不同的演算法使用相同的密鑰可能會洩漏有關密鑰的信息。因此，建議將密鑰限制為單個演算法。
- 使用“direct”作為接收者演算法並結合第二個接收者演算法將直接密鑰暴露給第二個接收者。
- 本文中的一些演算法限制密鑰的使用次數，而不會洩漏有關密鑰的信息。

使用 ECDH 和直接加 KDF（沒有密鑰包裝）不會直接導致私鑰洩漏； KDF 的單向功能可以防止這種情況發生。但是，有一個不同的問題需要解決。擁有兩個接收者需要在兩個接收者之間共享 CEK。因此，第二個接收者具有 CEK，該 CEK 源自可用於弱原始證明的材料。第二個接收者可以使用相同的 CEK 建立訊息並將其發送給第一個接收者;對於靜態 - 靜態 ECDH 或直接加 KDF，第一個接收者會假設 CEK 可以用於原始證明，即使它來自錯誤的實體。如果添加密鑰換行步驟，則不會暗示原始證明，這不是問題。

雖然之前已經提到過，但在某些情況下，已經證明使用單一密鑰進行多種演算法洩漏關於密鑰的資訊，為攻擊者提供偽造完整性標籤或獲取有關加密內容的資訊的機會。將密鑰綁定到單一演算法可以防止出現這些問題。強烈建議密鑰建立者和密鑰消費者不僅要為每種不同的演算法建立新密鑰，而且要在任何密鑰材料分佈中包含該演算法選擇，並嚴格執行密鑰結構中的演算法與訊息

結構中的演算法的匹配。除了檢查演算法是否正確外，還需要檢查密鑰形式。不要在需要“OKP”鍵值的地方使用“EC2”鍵值。

在使用密鑰進行傳輸之前，或者在對收到的資訊採取行動之前，需要對密鑰作出信任決定。與密鑰關聯的實體有權查看或請求權利的資料或操作是什麼？許多因素與此信任決策相關。這裡強調的一些是：

- 與密鑰所有者相關的權限是什麼？
- 加密演算法在當前上下文中是否可接受？
- 是否檢查與密鑰相關的限制，例如演算法或新鮮度，它們是否正確？
- 鑑於應用程式的當前狀態，請求是否合理？
- 是否已強制執行作為訊息一部分的任何安全注意事項（由應用程式或“crit”參數指定）？

本文中提供大量使用隨機數值的演算法。對於本文中定義的所有隨機數，對隨機數的某些型態的限制是對於鍵值或某些其他條件的唯一值。在所有這些情況下，沒有必要要求隨機數既獨特又不可預測；在這種情況下，使用計數器建立隨機數是合理的。在希望隨機數的模式不可預測以及唯一的情況下，可以使用為此目的建立的密鑰並加密計數器以產生隨機數值。

一個開始曝露的領域是根據訊息的長度對加密訊息進行流量分析。該規範沒有提供提供填充作為訊息結構的一部分的統一方法。觀察者可以基於本文中定義的所有內容加密演算法的長度來區分兩個不同的字串（例如，“是”和“否”）。這意味著由應用程式來記錄如何進行內容填充以防止或阻止此類分析。（例如，字串可以定義為“YES”和“NO”。）

18. 參考文獻

18.1. 規範性參考文獻

[AES-GCM] National Institute of Standards and Technology,

"Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST Special Publication 800-38D, DOI 10.6028/NIST.SP.800-38D,November 2007, <https://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf>.

[COAP.Formats] IANA, "Constrained RESTful Environments(CoRE) Parameters", <http://www.iana.org/assignments/core-parameters/>.

[DSS] National Institute of Standards and Technology, "Digital Signature Standard (DSS)", FIPS PUB 186-4, DOI 10.6028/NIST.FIPS.186-4, July 2013, <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>.

[MAC] National Institute of Standards and Technology, "Computer Data Authentication", FIPS PUB 113, May 1985, <http://csrc.nist.gov/publications/fips/fips113/fips113.html>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC:Keyed- Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <http://www.rfc-editor.org/info/rfc2104>.

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

[RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, DOI10.17487/RFC3394,

September 2002,
<http://www.rfc-editor.org/info/rfc3394>.

[RFC3610] Whiting, D., Housley, R., and N. Ferguson, "Counter with CBC-MAC (CCM)", RFC 3610, DOI 10.17487/RFC3610,September 2003, <http://www.rfc-editor.org/info/rfc3610>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <http://www.rfc-editor.org/info/rfc5869>.

[RFC6090] McGrew, D., Igoe, K., and M. Salter, "FundamentalElliptic Curve Cryptography Algorithms", RFC 6090, DOI 10.17487/RFC6090, February 2011, <http://www.rfc-editor.org/info/rfc6090>.

[RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, DOI 10.17487/RFC6979,August 2013, <http://www.rfc-editor.org/info/rfc6979>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI10.17487/RFC7049, October 2013, <http://www.rfc-editor.org/info/rfc7049>.

[RFC7539] Nir, Y. and A. Langley, "ChaCha20 and Poly1305 forIETF Protocols", RFC 7539, DOI 10.17487/RFC7539, May2015, <http://www.rfc-editor.org/info/rfc7539>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "EllipticCurves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <http://www.rfc-editor.org/info/rfc7748>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <http://www.rfc-editor.org/info/rfc8032>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI10.17487/RFC8174, May 2017, <http://www.rfc-editor.org/info/rfc8174>.

[SEC1] Certicom Research, "SEC 1: Elliptic Curve Cryptography", Standards for Efficient Cryptography, Version 2.0, May 2009, <http://www.secg.org/sec1-v2.pdf>.

18.2. 資訊參考

[CDDL] Vigano, C. and H. Birkholz, "CBOR data definitionlanguage (CDDL): a notational convention to express CBOR data structures", Work in Progress,draft-greevenbosch-appsawg-cbor-cddl-09, March 2017.

[OSCOAP] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security of CoAP (OSCOAP)", Work in Progress, draft-ietf-core-object-security-03, May 2017.

[PVSig] Brown, D. and D. Johnson, "Formal Security Proofs fora Signature Scheme with Partial Message Recovery", DOI 10.1007/3-540-45353-9_11, LNCS Volume 2020, June2000.

[RFC2633] Ramsdell, B., Ed., "S/MIME Version 3 Message Specification", RFC 2633, DOI 10.17487/RFC2633, June1999, <http://www.rfc-editor.org/info/rfc2633>.

[RFC4231] Nystrom, M., "Identifiers and Test Vectors for HMAC-SHA- 224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512", RFC 4231, DOI 10.17487/RFC4231, December 2005, <http://www.rfc-editor.org/info/rfc4231>.

[RFC4262] Santesson, S., "X.509 Certificate Extension for Secure/ Multipurpose Internet Mail Extensions (S/MIME) Capabilities", RFC 4262, DOI 10.17487/RFC4262,December 2005, <http://www.rfc-editor.org/info/rfc4262>.

[RFC4493] Song, JH., Poovendran, R., Lee, J., and T. Iwata, "The AES-CMAC Algorithm", RFC 4493, DOI 10.17487/RFC4493,June 2006, <http://www.rfc-editor.org/info/rfc4493>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August2007, <http://www.rfc-editor.org/info/rfc4949>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January2008, <http://www.rfc-editor.org/info/rfc5116>.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T.Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March2009, <http://www.rfc-editor.org/info/rfc5480>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <http://www.rfc-editor.org/info/rfc5652>.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/MultipurposeInternet Mail Extensions

(S/MIME) Version 3.2 Message Specification", RFC 5751, DOI 10.17487/RFC5751, January 2010, <http://www.rfc-editor.org/info/rfc5751>.

[RFC5752] Turner, S. and J. Schaad, "Multiple Signaturesin Cryptographic Message Syntax (CMS)", RFC 5752, DOI 10.17487/RFC5752, January 2010, <http://www.rfc-editor.org/info/rfc5752>.

[RFC5990] Randall, J., Kaliski, B., Brainard, J., and S. Turner, "Use of the RSA-KEM Key Transport Algorithm in the Cryptographic Message Syntax (CMS)", RFC 5990,DOI 10.17487/RFC5990, September 2010, <http://www.rfc-editor.org/info/rfc5990>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <http://www.rfc-editor.org/info/rfc6151>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <http://www.rfc-editor.org/info/rfc6838>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159,March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252,DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI

10.17487/RFC7515,May 2015,
<http://www.rfc-editor.org/info/rfc7515>.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption(JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <http://www.rfc-editor.org/info/rfc7516>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC7517, DOI 10.17487/RFC7517, May 2015, <http://www.rfc-editor.org/info/rfc7517>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC7518, DOI 10.17487/RFC7518, May 2015, <http://www.rfc-editor.org/info/rfc7518>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <http://www.rfc-editor.org/info/rfc8017>.

[RFC8018] Moriarty, K., Ed., Kaliski, B., and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version2.1", RFC 8018, DOI 10.17487/RFC8018, January 2017, <http://www.rfc-editor.org/info/rfc8018>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <http://www.rfc-editor.org/info/rfc8126>.

[SP800-56A]Barker, E., Chen, L., Roginsky, A., and M. Smid, "Recommendation for Pair-Wise Key EstablishmentSchemes Using Discrete Logarithm Cryptography", NIST Special Publication 800-56A, Revision 2,DOI 10.6028/NIST.SP.800-56Ar2, May 2013,

<http://nvlpubs.nist.gov/nistpubs/SpecialPublications/ NIST.SP.800-56Ar2.pdf>.

[W3C.WebCrypto]
Watson, M., "Web Cryptography API", W3C Recommendation, January 2017, <https://www.w3.org/TR/WebCryptoAPI/>.

附錄A 演算法外部資料認證指南

工作組的一部分表達放寬演算法識別符要求出現在 COSE 物件的每個級別中的規則的強烈願望。為支援這一立場，已經提出兩個基本原因。首先，如果從 CoAP 環境中的最常見訊息中省略演算法識別符，則結果訊息將更小。其次，如果在可以放置演算法識別符的不同位置之間沒有正確地進行完全檢查（訊息本身，應用程式語句，發送者擁有的密鑰結構，以及接收者擁有的密鑰結構），則會出現潛在的錯誤。

本附錄列出如何進行此類更改以及應用程式需要指定的詳細資訊才能使用此選項。指定兩組不同的細節：省略演算法識別符所需的細節以及在計數器簽章屬性上使用不包含自身屬性的變量所需的細節。

A.1. 演算法辨識

在本節中，列出三組建議。第一組建議適用於為 COSE 物件的單層辨識隱性演算法。第二組建議適用於為 COSE 物件的多層辨識多個隱性演算法。第三組建議適用於具有多個 COSE 物件構造的隱性演算法。

RFC 2119 的關鍵詞在這裡故意不使用。此規範可以提供建議，但不能強制執行。
這組建議適用於應用程式正在分配固定演算法以及用於單個 COSE 物件的密鑰資訊的情況。這通常適用於最小的 COSE 物件，特別是 COSE_Sign1，COSE_Mac0 和 COSE_Encrypt0，但也可以應用於其他結構。

應考慮以下項目：

- 應用程式需要列出要使用隱性演算法的 COSE 結構集。應用程式需要要求在這些結構之一中接收顯性演算法識別符將導致訊息被拒絕。這個要求被表明敘述，以便永遠不會出現關於應該使用哪種演算法的問題，隱性或顯性演算法的任何模糊性的情況。即使傳送的演算法識別符是受保護的屬性，這也適用。即使傳送的演算法與隱性演算法相同，這也適用。
- 當省略演算法識別符時，應用程式需要定義要被視為上下文一部分的資訊集。至少，這將是密鑰識別符（如果需要），密鑰，演算法和它使用的 COSE 結構。應用程式應將單一密鑰的使用限制為單個演算法。如本文中的一些演算法所述，在不同的相關演算法中使用相同的密鑰可能導致關於密鑰的信息洩漏，關於資料的洩漏或執行偽造的能力。
- 在許多情況下，使演算法識別符隱性的應用程式也會出於同樣的原因而隱含上下文識別符。也就是說，省略上下文識別符將減小訊息大小（可能顯著地取決於識別符的長度）。執行此操作的應用程式將需要描述要省略上下文識別符的情況以及在這些情況下如何推斷上下文識別符。（對所有的密鑰徹底搜索通常不被認為是可接受的。）如何做到這一點的一個例子是將上下文綁定到交易識別符。兩者都將在原始訊息上發送，但在該點之後僅需要發送交易識別符，因為上下文被綁定到交易識別符中。另一種方法是將上下文與網路位址相關聯。可以假設來自單一網路位址的所有訊息與特定上下文相關聯。（在這種情況下，地址通常會作為上下文的一部分進行分配。）
- 應用程式不能依賴密鑰識別符是唯一的，除非他們需要付出巨大努力來確保以建立此保證的方式計算密鑰識別符。即使應用程式執行此操作，如果應用程式在不同的上下文中運行（即，使用不同的上下文提供程序），或者如果系統將來自不同應用程式的安全上下文組合在一起，則可能違反唯一性。
- 應用程式應繼續保護演算法識別符的做法。由於這不是通過將其放在受保護的屬性欄位中來完成的，因此應用程式應定義包含此值的特定於應用程式的外部資料結構。該外部資料欄位可以用於內容加密，MAC 和簽章演算法。它可以在

SuppPrivInfo 欄位中用於那些使用 KDF 衍生鍵值的演算法。應用程式也可能希望保護作為上下文結構一部分的其他資訊。應當注意，那些欄位，例如密鑰或基本 IV，由於在加密計算中使用而受到保護，並且不需要包括在外部資料欄位中。

- 第二種情況是為多層 COSE 物件指定多個隱性演算法識別符。這將如何工作的範例是應用程式指定的加密上下文，其包含內容加密演算法，密鑰包裝演算法，密鑰識別符和共用密碼。發送者省略為內容層和接收者層發送演算法識別符，只留下密鑰識別符。接收器然後使用密鑰識別符來獲得隱性演算法識別符。

需要考慮以下附加項目：

- 想要支援此功能的應用程式需要定義一個結構，該結構允許並清楚地標識要與給定密鑰一起使用的 COSE 結構以及要用於輔助層的結構和演算法。從接收層開始，正常計算輔助層的密鑰。

第三種情況是具有多個隱性演算法識別符，但針對可能不相關的層或不同的 COSE 物件。有許多不同的情況可能適用。其中一些情況是：

- 兩個上下文成對分配。每個上下文都與 COSE_Encrypt 訊息一起使用。每個上下文將由不同的密鑰和 IV 組成，甚至可能包含不同的演算法。一個上下文用於從 A 方向 B 方發送訊息，第二個上下文用於從 B 方向 A 方發送訊息。這意味著當各方使用不同的密鑰發送它的訊息時，不會發生反射攻擊。；反射回它的訊息將無法解密。
- 兩個上下文成對分配。第一個上下文用於加密訊息，第二個上下文用於在訊息上放置計數器簽章。目的是第二上下文可以獨立於第一上下文分配給其他實體。這允許這些實體驗證訊息來自個人，而無法解密訊息並查看內容。
- 兩個上下文成對分配。第一個上下文包含用於處理 MACed 訊息的密鑰，第二個上下文包含用於處理加密訊息的密鑰。這允許向具有不同密鑰的不同型態的訊息的參與者統一分配密鑰，但是密鑰可以以協調的方式使用。

對於這些情況，需要考慮以下附加項目：

- 應用程式需要確保多個上下文保持關聯。如果其中一個上下文因任何原因而失效，則與其關聯的所有上下文也應該無效。

A.2. 沒有標頭的計數器簽章

有一個團體希望擁有一個與正在簽章的值直接相關的計數器簽章參數，因此可以從正在發送的訊息中刪除經過身份驗證和未經身份驗證的儲存區。對此的關注是更小的大小，因為關於建立計數器簽章的過程的所有訊息都是隱含的而不是在訊息中明確地攜帶。這不僅包括如上所述的演算法識別符，還包括諸如密鑰標識之類的項，其始終在簽章結構外部。這意味著正在執行計數器簽章驗證的實體需要從上下文推斷使用哪個密鑰而不是顯性。這樣做的一種方法是假設來自特定通訊埠（或特定 URL）的所有資料都要由特定密鑰驗證。（請注意，這並不要求密鑰識別符是簽章值的一部分，因為它不用於加密目的。如果密鑰驗證計數器簽章，則應該假設與該密鑰關聯的實體產生簽章。）

在計算空的計數器簽章標頭的簽章時，使用第 4.4 節中定義的相同 Sig_structure。省略 sign_protected 欄位，因為此計數器簽章標頭中沒有受保護的標頭欄位。“CounterSignature0”的值放在 Sig_stucture 的上下文欄位中。

Name	Label	Value Type	Value	Description
CounterSignature0	9	bstr		Counter signature with implied signer and headers

表 27：CounterSignature0 的標頭參數

附錄B 接收者資訊的兩個層次

所有當前定義的接收者演算法階級僅使用 COSE_Encrypt 結構的兩個層。第一層是訊息內容，第二層是內容密鑰加密。但是，如果使用諸如 RSA 密鑰封裝機制（RSA-KEM）之類的接收者演算法（參見 RSA-KEM [RFC5990]的附錄 A），則有三層 COSE_Encrypt 結構是有意義的。

這些層將是：

- 第 0 層：內容加密層。該層包含訊息的負載。
- 第 1 層：KEK 對 CEK 的加密。
- 第 2 層：使用 RSA 密鑰和密鑰衍生函數對長隨機密鑰進行加密，以將該隱密值轉換為 KEK。

這是三層訊息例子的樣子。

該訊息具有以下層：

- 第 0 層：使用 128 位元密鑰使用 AES-GCM 加密內容。
- 第 1 層：使用帶有 128 位元密鑰的 AES 密鑰包裝演算法。
- 第 2 層：使用 ECDH 短暫靜態直接生成第 1 層密鑰。

實際上，此範例是使用 ECDH-ES + A128KW 演算法的分解版本。

二進制文件的大小為 183 個位元組

```
96(
  [
    / protected / h'a10101' / {
        \ alg \ 1:1 \ AES-GCM 128 \
      } / ,
    / unprotected / {
      / iv / 5:h'02d1f7e6f26c43d4868d87ce'
    },
    / ciphertext / h'64f84d913ba60a76070a9a48f26e97e863e2852948658f0
811139868826e89218a75715b',
    / recipients / [
      [
        / protected / h'',
        / unprotected / {
          / alg / 1:-3 / A128KW /
        },
        / ciphertext / h'dbd43c4e9d719c27c6275c67d628d493f090593db82
18f11',
        / recipients / [
          [
            / protected / h'a1013818' / {
                \ alg \ 1:-25 \ ECDH-ES + HKDF-256 \
              } / ,
            / unprotected / {
              / ephemeral / -1:{
                / kty / 1:2,
                / crv / -1:1,
                / x / -2:h'b2add44368ea6d641f9ca9af308b4079aeb519f11
e9b8a55a600b21233e86e68',
                / y / -3:false
              },
              / kid / 4:'meriadoc.brandybuck@buckland.example'
            },
            / ciphertext / h''
          ]
        ]
      ]
    ]
  ]
)
```

附錄C 範例

本附錄包含一組範例，這些範例顯示本文中定義的不同功能和訊息型態。為了使範例更易於閱讀，它們使用擴充的 CBOR 診斷表示法（在[CDDL]中定義）而不是二進制傾印表示。

已經在<https://github.com/cose-wg/ Examples>建立一個 GitHub 項目，該項目不僅包含本文中提供的範例，還包含更完整的測試範例集。每個範例都在一個 JSON 文件中找到，該文包含用於建立範例的輸入，一些可用於調試範例的中間值以及以十六進制和

CBOR 診斷表示法格式顯示的範例的輸出。該位置的一些範例是設計故障測試案例；這些在 JSON 文件中明確標示。如果找到本文中範例中的錯誤，將更新 GitHub 上的範例，並將在 JSON 文件中放置相應的註釋。

如上所述，這些範例使用 CBOR 的診斷表示法表示。存在一個基於 Ruby 的工具，可以在診斷表示法和二進制文件之間進行轉換。可以使用命令行安裝此工具：

可以使用以下命令行將診斷表示法轉換為二進制文件：

可以通過 XPath 表達式從本文的 XML 版本中提取範例，因為所有作品都使用屬性 type = 'CBORdiag' 進行標籤。（根據正在使用的 XPath 評估程序，可能需要處理＆gt；作為實體。）

//artwork[@type='CDDL']/text()

C.1. 簽章訊息的範例

C.1.1. 單一簽章

此範例使用以下內容：

- 簽章演算法：ECDSA w / SHA-256，曲線 P-256

二進制文件的大小為 103 位元組

```
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'
      ]
    ]
  ]
)
```

C.1.2. 多個簽章者

此範例使用以下內容：

- 簽章演算法：ECDSA w / SHA-256，曲線 P-256
- 簽章演算法：ECDSA w / SHA-512，曲線 P-521

二進制文件的大小是 277 個位元組

```
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'
      ],
      [
        / protected / h'a1013823' / {
            \ alg \ 1:-36
          } / ,
        / unprotected / {
          / kid / 4:'bilbo.baggins@hobbiton.example'
        },
        / signature / h'00a2d28a7c2bdb1587877420f65adf7d0b9a06635dd1
de64bb62974c863f0b160dd2163734034e6ac003b01e8705524c5c4ca479a952f024
7ee8cb0b4fb7397ba08d009e0c8bf482270cc5771aa143966e5a469a09f613488030
c5b07ec6d722e3835adb5b2d8c44e95ffb13877dd2582866883535de3bb03d01753f
83ab87bb4f7a0297'
      ]
    ]
  ]
)
```

C.1.3. 反簽章

此範例使用以下內容：

- 簽章演算法：ECDSA w / SHA-256，曲線 P-256
- 簽章和計數器簽章使用相同的參數。

二進制文件的大小是 180 位元組

```
98(
  [
    / protected / h'',
    / unprotected / {
      / countersign / 7:[
        / protected / h'a10126' / {
            \ alg \ 1:-7 \ ECDSA 256 \
          } / ,
        / unprotected / {
          / kid / 4:'11'
        },
        / signature / h'5ac05e289d5d0e1b0a7f048a5d2b643813ded50bc9e4
9220f4f7278f85f19d4a77d655c9d3b51e805a74b099e1e085aacd97fc29d72f887e
8802bb6650cceb2c'
      ]
    },
    / payload / 'This is the content.',
    / signatures / [
      [
        / protected / h'a10126' / {
            \ alg \ 1:-7 \ ECDSA 256 \
          } / ,
        / unprotected / {
          / kid / 4:'11'
        },
        / signature / h'e2aeafd40d69d19dfe6e52077c5d7ff4e408282cbefb
5d06cbf414af2e19d982ac45ac98b8544c908b4507de1e90b717c3d34816fe926a2b
98f53afd2fa0f30a'
      ]
    ]
  ]
)
```

C.1.4. 具有重要性的簽章

此範例使用以下內容：

- 簽章演算法：ECDSA w / SHA-256，曲線 P-256
- "reserved"標頭參數上有一個關鍵性標籤

二進制文件的大小是 125 個位元組

```
98(
  [
    / protected / h'a2687265736572766564f4028168726573657276656 4' /
{
        "reserved":false,
         \ crit \ 2:[
           "reserved"
         ]
       } / ,
     / unprotected / {},
     / payload / 'This is the content.',
     / signatures / [
       [
         / protected / h'a10126' / {
             \ alg \ 1:-7 \ ECDSA 256 \
           } / ,
         / unprotected / {
           / kid / 4:'11'
         },
         / signature / h'3fc54702aa56e1b2cb20284294c9106a63f91bac658d
69351210a031d8fc7c5ff3e4be39445b1a3e83e1510d1aca2f2e8a7c081c7645042b
18aba9d1fad1bd9c'
       ]
    ]
  ]
)
```

C.2. 單一簽章者範例

C.2.1. 單個 ECDSA 簽章

此範例使用以下內容：

- 簽章演算法：ECDSA w / SHA-256，曲線 P-256

二進制文件的大小是 98 個位元組

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

C.3. 已封閉訊息的範例

C.3.1. 直接 ECDH

此範例使用以下內容：

- CEK：AES-GCM w / 128 位元密鑰
- 接收者類別：ECDH Ephemeral-Static，Curve P-256

二進制文件的大小為 151 個位元組

```
96(
  [
    / protected / h'a10101' / {
        \ alg \ 1:1 \ AES-GCM 128 \
      } / ,
    / unprotected / {
      / iv / 5:h'c9cf4df2fe6c632bf7886413'
    },
    / ciphertext / h'7adbe2709ca818fb415f1e5df66f4e1a51053ba6d65a1a0
c52a357da7a644b8070a151b0',
    / recipients / [
      [
        / protected / h'a1013818' / {
            \ alg \ 1:-25 \ ECDH-ES + HKDF-256 \
          } / ,
        / unprotected / {
          / ephemeral / -1:{
            / kty / 1:2,
            / crv / -1:1,
            / x / -2:h'98f50a4ff6c05861c8860d13a638ea56c3f5ad7590bbf
bf054e1c7b4d91d6280',
            / y / -3:true
          },
          / kid / 4:'meriadoc.brandybuck@buckland.example'
        },
        / ciphertext / h''
      ]
    ]
  ]
)
```

C.3.2. 直接加密鑰推導

此範例使用以下內容：

- CEK：AES-CCM w / 128 位元密鑰，將標籤截斷為 64 位元
- 接收者類別：在共用密碼上使用 HKDF，並將以下隱含欄位作為上下文的一部分。
  - salt：“aabbccddeeffgghh”
  - PartyU 識別：“照明客戶端”
  - PartyV 識別：“照明伺服器”

* 補充公共其他：“加密範例 02”

二進制文件的大小是 91 個位元組

```
96(
  [
    / protected / h'a1010a' / {
       \ alg \ 1:10 \ AES-CCM-16-64-128 \
      } / ,
    / unprotected / {
      / iv / 5:h'89f52f65a1c580933b5261a76c'
    },
    / ciphertext / h'753548a19b1307084ca7b2056924ed95f2e3b17006dfe93
1b687b847',
    / recipients / [
      [
        / protected / h'a10129' / {
           \ alg \ 1:-10
          } / ,
        / unprotected / {
          / salt / -20:'aabbccddeeffgghh',
          / kid / 4:'our-secret'
        },
        / ciphertext / h''
      ]
    ]
  ]
```

C.3.3. 加密內容的計數器簽章

此範例使用以下內容：

- CEK：AES-GCM w / 128 位元密鑰
- 接收者類別：ECDH Ephemeral-Static，Curve P-256

二進制文件的大小為 326 位元組

```
96(
  [
    / protected / h'a10101' / {
       \ alg \ 1:1 \ AES-GCM 128 \
      } / ,
    / unprotected / {
      / iv / 5:h'c9cf4df2fe6c632bf7886413',
      / countersign / 7:[
        / protected / h'a1013823' / {
           \ alg \ 1:-36
          } / ,
        / unprotected / {
          / kid / 4:'bilbo.baggins@hobbiton.example'
        },
        / signature / h'00929663c8789bb28177ae28467e66377da12302d7f9
594d2999afa5dfa531294f8896f2b6cdf1740014f4c7f1a358e3a6cf57f4ed6fb02f
cf8f7aa989f5dfd07f0700a3a7d8f3c604ba70fa9411bd10c2591b483e1d2c31de00
3183e434d8fba18f17a4c7e3dfa003ac1cf3d30d44d2533c4989d3ac38c38b71481c
c3430c9d65e7ddff'
      ]
    },
    / ciphertext / h'7adbe2709ca818fb415f1e5df66f4e1a51053ba6d65a1a0
c52a357da7a644b8070a151b0',
    / recipients / [
      [
        / protected / h'a1013818' / {
           \ alg \ 1:-25 \ ECDH-ES + HKDF-256 \
          } / ,
        / unprotected / {
          / ephemeral / -1:{
            / kty / 1:2,
            / crv / -1:1,
            / x / -2:h'98f50a4ff6c05861c8860d13a638ea56c3f5ad7590bbf
bf054e1c7b4d91d6280',
            / y / -3:true
          },
          / kid / 4:'meriadoc.brandybuck@buckland.example'
        },
        / ciphertext / h''
      ]
    ]
  ]
)
```

C.3.4. 使用外部資料加密的內容

此範例使用以下內容：

- CEK：AES-GCM w / 128 位元密鑰
- 接收者類：ECDH static-Static，Curve P-256 使用 AES 密鑰包裝
- 外部供應的 AAD：h'0011bbcc22dd44ee55ff660077'

二進制文件的大小為 173 位元組

```
96(
  [
    / protected / h'a10101' / {
       \ alg \ 1:1 \ AES-GCM 128 \
      } / ,
    / unprotected / {
      / iv / 5:h'02d1f7e6f26c43d4868d87ce'
    },
    / ciphertext / h'64f84d913ba60a76070a9a48f26e97e863e28529d8f5335
e5f0165eee976b4a5f6c6f09d',
    / recipients / [
      [
        / protected / h'a101381f' / {
           \ alg \ 1:-32 \ ECHD-SS+A128KW \
          } / ,
        / unprotected / {
          / static kid / -3:'peregrin.took@tuckborough.example',
          / kid / 4:'meriadoc.brandybuck@buckland.example',
          / U nonce / -22:h'0101'
        },
        / ciphertext / h'41e0d76f579dbd0d936a662d54d8582037de2e366fd
e1c62'
      ]
    ]
  ]
)
```

C.4. 加密訊息的範例

C.4.1. 簡單的加密訊息

此範例使用以下內容：

- CEK：AES-CCM，帶 128 位元密鑰和 64 位元標籤

二進制文件的大小是 52 個位元組

```
16(
  [
    / protected / h'a1010a' / {
       \ alg \ 1:10 \ AES-CCM-16-64-128 \
      } / ,
    / unprotected / {
      / iv / 5:h'89f52f65a1c580933b5261a78c'
    },
    / ciphertext / h'5974e1b99a3a4cc09a659aa2e9e7fff161d38ce71cb45ce
460ffb569'
  ]
)
```

C.4.2. 具有部分 IV 的加密訊息

此範例使用以下內容：

- CEK：AES-CCM，帶 128 位元密鑰和 64 位元標籤
- IV 的前綴是 89F52F65A1C580933B52

二進制文件的大小為 41 個位元組

```
16(
  [
    / protected / h'a1010a' / {
        \ alg \ 1:10 \ AES-CCM-16-64-128 \
      } / ,
    / unprotected / {
      / partial iv / 6:h'61a7'
    },
    / ciphertext / h'252a8911d465c125b6764739700f0141ed09192de139e05
3bd09abca'
  ]
)
```

C.5. MAC 訊息的範例

C.5.1. 共用密碼直接 MAC

此範例使用以下內容：

- MAC：AES-CMAC，256 位元密鑰，截斷為 64 位元
- 接收者類別：直接共用密碼

二進制文件的大小是 57 個位元組

```
97(
  [
    / protected / h'a1010f' / {
        \ alg \ 1:15 \ AES-CBC-MAC-256//64 \
      } / ,
    / unprotected / {},
    / payload / 'This is the content.',
    / tag / h'9e1226ba1f81b848',
    / recipients / [
      [
        / protected / h'',
        / unprotected / {
          / alg / 1:-6 / direct /,
          / kid / 4:'our-secret'
        },
        / ciphertext / h''
      ]
    ]
  ]
)
```

C.5.2. CECDH 直接 MAC

此範例使用以下內容：

- MAC：HMAC w / SHA-256，256 位元密鑰
- 接收者類別：ECDH 密鑰協定，兩個靜態密鑰，HKDF w /上下文結構

二進制文件的大小是 214 個位元組

```
97(
  [
    / protected / h'a10105' / {
        \ alg \ 1:5 \ HMAC 256//256 \
      } / ,
    / unprotected / {},
    / payload / 'This is the content.',
    / tag / h'81a03448acd3d305376eaa11fb3fe416a955be2cbe7ec96f012c99
4bc3f16a41',
    / recipients / [
      [
        / protected / h'a101381a' / {
            \ alg \ 1:-27 \ ECDH-SS + HKDF-256 \
          } / ,
        / unprotected / {
          / static kid / -3:'peregrin.took@tuckborough.example',
          / kid / 4:'meriadoc.brandybuck@buckland.example',
          / U nonce / -22:h'4d8553e7e74f3c6a3a9dd3ef286a8195cbf8a23d
19558ccfec7d34b824f42d92bd06bd2c7f0271f0214e141fb779ae2856abf585a583
68b017e7f2a9e5ce4db5'
        },
        / ciphertext / h''
      ]
    ]
  ]
)
```

C.5.3. 包裝的 MAC

此範例使用以下內容：

- MAC：AES-MAC，128 位元密鑰，截斷為 64 位元
- 接收者類別：帶有預先共享 256 位元密鑰的 AES 密鑰包裝

二進制文件的大小為 109 個位元組

```
97(
  [
    / protected / h'a1010e' / {
       \ alg \ 1:14 \ AES-CBC-MAC-128//64 \
      } / ,
    / unprotected / {},
    / payload / 'This is the content.',
    / tag / h'36f5afaf0bab5d43',
    / recipients / [
      [
        / protected / h'',
        / unprotected / {
          / alg / 1:-5 / A256KW /,
          / kid / 4:'018c0ae5-4d9b-471b-bfd6-eef314bc7037'
        },
        / ciphertext / h'711ab0dc2fc4585dce27effa6781c8093eba906f227
b6eb0'
      ]
    ]
  ]
)
```

C.5.4. 多接收者 MACed 訊息

此範例使用以下內容：

- MAC：HMAC w / SHA-256,128 位元密鑰
- 接收者類別：使用三種不同的方法
  1. ECDH Ephemeral-Static，Curve P-521，帶有 128 位元密鑰的 AES 密鑰包裝
  2. 帶有 256 位元密鑰的 AES 密鑰包裝

二進制文件的大小為 309 位元組

```
97(
  [
    / protected / h'a10105' / {
       \ alg \ 1:5 \ HMAC 256//256 \
      } / ,
    / unprotected / {},
    / payload / 'This is the content.',
    / tag / h'bf48235e809b5c42e995f2b7d5fa13620e7ed834e337f6aa43df16
1e49e9323e',
    / recipients / [
      [
        / protected / h'a101381c' / {
           \ alg \ 1:-29 \ ECHD-ES+A128KW \
          } / ,
        / unprotected / {
          / ephemeral / -1:{
            / kty / 1:2,
            / crv / -1:3,
            / x / -2:h'0043b12669acac3fd27898ffba0bcd2e6c366d53bc4db
71f909a759304acfb5e18cdc7ba0b13ff8c7636271a6924b1ac63c02688075b55ef2
d613574e7dc242f79c3',
            / y / -3:true
          },
          / kid / 4:'bilbo.baggins@hobbiton.example'
        },
        / ciphertext / h'339bc4f79984cdc6b3e6ce5f315a4c7d2b0ac466fce
a69e8c07dfbca5bb1f661bc5f8e0df9e3eff5'
      ],
      [
        / protected / h'',
        / unprotected / {
          / alg / 1:-5 / A256KW /,
          / kid / 4:'018c0ae5-4d9b-471b-bfd6-eef314bc7037'
        },
        / ciphertext / h'0b2c7cfce04e98276342d6476a7723c090dfdd15f9a
518e7736549e998370695e6d6a83b4ae507bb'
      ]
    ]
  ]
)
```

C.6. MAC0 訊息的範例

C.6.1. 共用密碼直接 MAC

此範例使用以下內容：

- MAC：AES-CMAC，256 位元密鑰，截斷為 64 位元
- 接收者類別：直接共用密碼

二進制文件的大小是 37 個位元組

```
17(
  [
    / protected / h'a1010f' / {
        \ alg \ 1:15 \ AES-CBC-MAC-256//64 \
      } / ,
    / unprotected / {},
    / payload / 'This is the content.',
    / tag / h'726043745027214f'
  ]
)
```

請注意，此範例使用與附錄 C.5.1 相同的輸入。

C.7. COSE 鍵值

C.7.1. 公鑰

這是 COSE 密鑰集的範例。此範例包含所有前面範例的公鑰。

按密鑰為：

- 帶有“meriadoc.brandybuck@buckland.example”的 EC 密鑰
- 帶有“peregrin.took@tuckborough.example”的 EC 密鑰
- 帶有“bilbo.baggins@hobbiton.example”的 EC 密鑰
- 帶有“11”的 EC 密鑰

二進制文件的大小是 481 個位元組

```
[
  {
    -1:1,
    -2:h'65eda5a12577c2bae829437fe338701a10aaa375e1bb5b5de108de439c0
8551d',
    -3:h'1e52ed75701163f7f9e40ddf9f341b3dc9ba860af7e0ca7ca7e9eecd008
4d19c',
    1:2,
    2:'meriadoc.brandybuck@buckland.example'
  },
  {
    -1:1,
    -2:h'bac5b11cad8f99f9c72b05cf4b9e26d244dc189f745228255a219a86d6a
09eff',
    -3:h'20138bf82dc1b6d562be0fa54ab7804a3a64b6d72ccfed6b6fb6ed28bbf
c117e',
    1:2,
    2:'11'
  },
  {
    -1:3,
    -2:h'0072992cb3ac08ecf3e5c63dedec0d51a8c1f79ef2f82f94f3c737bf5de
7986671eac625fe8257bbd0394644caaa3aaf8f27a4585fbbcad0f2457620085e5c8
f42ad',
    -3:h'01dca6947bce88bc5790485ac97427342bc35f887d86d65a089377e247e
60baa55e4e8501e2ada5724ac51d6909008033ebc10ac999b9d7f5cc2519f3fe1ea1
d9475',
    1:2,
    2:'bilbo.baggins@hobbiton.example'
  },
  {
    -1:1,
    -2:h'98f50a4ff6c05861c8860d13a638ea56c3f5ad7590bbfbf054e1c7b4d91
d6280',
    -3:h'f01400b089867804b8e9fc96c3932161f1934f4223069170d924b7e03bf
822bb',
    1:2,
    2:'peregrin.took@tuckborough.example'
  }
]
```

C.7.2. 私鑰

這是 COSE 密鑰集的範例。此範例包含所有前面範例的私鑰。按密鑰為：

- 帶有“meriadoc.brandybuck@buckland.example”的 EC 密鑰
- 帶有“our-secret”的共用密碼
- 帶有“peregrin.took@tuckborough.example”的 EC 密鑰
- 帶有“018c0ae5-4d9b-471bbfd6- eef314bc7037”的共用密碼
- 帶有“bilbo.baggins@hobbiton.example”的 EC 密鑰

- 帶有“11”的 EC 密鑰

二進制文件的大小為 816 位元組

```
[
  {
    1:2,
    2:'meriadoc.brandybuck@buckland.example',
    -1:1,
    -2:h'65eda5a12577c2bae829437fe338701a10aaa375e1bb5b5de108de439c0
8551d',
    -3:h'1e52ed75701163f7f9e40ddf9f341b3dc9ba860af7e0ca7ca7e9eecd008
4d19c',
    -4:h'aff907c99f9ad3aae6c4cdf21122bce2bd68b5283e6907154ad911840fa
208cf'
  },
  {
    1:2,
    2:'11',
    -1:1,
    -2:h'bac5b11cad8f99f9c72b05cf4b9e26d244dc189f745228255a219a86d6a
09eff',
    -3:h'20138bf82dc1b6d562be0fa54ab7804a3a64b6d72ccfed6b6fb6ed28bbf
c117e',
    -4:h'57c92077664146e876760c9520d054aa93c3afb04e306705db609030850
7b4d3'
  },
  {
    1:2,
    2:'bilbo.baggins@hobbiton.example',
    -1:3,
    -2:h'0072992cb3ac08ecf3e5c63dedec0d51a8c1f79ef2f82f94f3c737bf5de
7986671eac625fe8257bbd0394644caaa3aaf8f27a4585fbbcad0f2457620085e5c8
f42ad',
    -3:h'01dca6947bce88bc5790485ac97427342bc35f887d86d65a089377e247e
60baa55e4e8501e2ada5724ac51d6909008033ebc10ac999b9d7f5cc2519f3fe1ea1
d9475',
    -4:h'00085138ddabf5ca975f5860f91a08e91d6d5f9a76ad4018766a476680b
55cd339e8ab6c72b5facdb2a2a50ac25bd086647dd3e2e6e99e84ca2c3609fdf177f
eb26d'
  },
  {
    1:4,
    2:'our-secret',
    -1:h'849b57219dae48de646d07dbb533566e976686457c1491be3a76dcea6c4
27188'
  },
 {
    1:2,
    -1:1,
     2:'peregrin.took@tuckborough.example',
    -2:h'98f50a4ff6c05861c8860d13a638ea56c3f5ad7590bbfbf054e1c7b4d91
d6280',
    -3:h'f01400b089867804b8e9fc96c3932161f1934f4223069170d924b7e03bf
822bb',
    -4:h'02d1f7e6f26c43d4868d87ceb2353161740aacf1f7163647984b522a848
df1c3'
  },
  {
    1:4,
```

```
    2:'our-secret2',
    -1:h'849b5786457c1491be3a76dcea6c4271'
  },
  {
    1:4,
    2:'018c0ae5-4d9b-471b-bfd6-eef314bc7037',
    -1:h'849b57219dae48de646d07dbb533566e976686457c1491be3a76dcea6c4
27188'
  }
]
```

致謝

本文是 IETF 的 COSE 工作組的產品。

首先要讓我開始這個責任項目是以下個人：Richard Barnes，Matt Miller 和 Martin Thomson。

規範的初始版本在某種程度上基於 JOSE 和 S / MIME 工作組的輸出。

以下人員對文件的最終形式提供與投入心力：Carsten Bormann，John Bradley，Brain Campbell，Michael B. Jones，Ilari Liusvaara，Francesca Palombini，Ludwig Seitz 和 Goran Selander。

作者資訊

Jim Schaad
August Cellars
電子郵件：ietf@augustcellars.com