ICS 35.030
CCS L 80

中华人民共和国密码行业标准

GM/T 0125.2—2022

JSON Web 密码应用语法规范 第 2 部分:数字签名

JavaScript Object Notation Web cryptographic application syntax specification—Part 2:Signature

2022-11-20 发布　　2023-06-01 实施

国家密码管理局　　发 布

目　　次

前言 …… Ⅰ
引言 …… Ⅱ
1　范围 …… 1
2　规范性引用文件 …… 1
3　术语和定义 …… 1
4　缩略语 …… 2
5　符号 …… 2
6　JSON Web 数字签名(JWS) …… 2
6.1　概述 …… 2
6.2　JOSE 头部 …… 2
7　JWS 数字签名算法 …… 4
7.1　概述 …… 4
7.2　SM2 数字签名算法 …… 4
8　JWS 消息鉴别算法 …… 5
8.1　概述 …… 5
8.2　基于 SM3 算法的消息鉴别算法 …… 5
9　JWS 生成和验证 …… 5
9.1　概述 …… 5
9.2　生成流程 …… 5
9.3　验证流程 …… 5
10　字符串比较规则 …… 6
11　密钥身份识别 …… 6
12　序列化 …… 6
12.1　概述 …… 6
12.2　紧凑序列化 …… 7
12.3　JSON 序列化 …… 7
附录 A（资料性）　JWS 示例 …… 9
A.1　综述 …… 9
A.2　SM2 算法对有效载荷进行签名示例 …… 9
A.3　SM2 算法对有效载荷生成消息鉴别码示例 …… 10
A.4　SM2 算法对有效载荷进行多人签名示例 …… 11

前　言

本文件按照GB/T 1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

本文件是GM/T 0125《JSON Web密码应用语法规范》的第2部分。GM/T 0125已经发布了以下部分：

——第1部分：算法标识；

——第2部分：数字签名；

——第3部分：数据加密；

——第4部分：密钥。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由密码行业标准化技术委员会提出并归口。

本文件起草单位：广东省电子商务认证有限公司、智巡密码（上海）检测技术有限公司、格尔软件股份有限公司、北京信安世纪科技股份有限公司、北京数字认证股份有限公司、北京国脉信安科技有限公司、中国科学院信息工程研究所、广东南方通信建设有限公司、上海市数字证书认证中心有限公司。

本文件主要起草人：陈树乐、韩玮、郑强、张永强、袁峰、高能、张庆勇、赵敏、刘义、黄志伟、林少柳、梁宁宁、梁家声、傅大鹏、傅鹏、王维初。

引　言

《JSON Web 密码应用语法规范》旨在以国产商用密码算法为核心，来保证数据机密性和完整性，适用于 JSON Web 密码应用产品的研发与检测，其他使用 JSON 数据交换格式的安全产品，可参考使用。《JSON Web 密码应用语法规范》由四个部分构成。

——第 1 部分：算法标识。定义了 JSON Web 密码应用的算法标识。

——第 2 部分：数字签名。描述了基于 JSON 数据结构来保护消息内容的数字签名或消息鉴别码的语法规范，并给出了相应的生成和验证流程。

——第 3 部分：数据加密。描述了使用身份鉴别和加密来确保数据的机密性和完整性的技术要求。

——第 4 部分：密钥。定义了密钥的 JSON 数据结构表示方法。

本文件为《JSON Web 密码应用语法规范》的第 2 部分，描述了基于 JSON 数据结构来保护消息内容的数字签名或消息鉴别码的语法规范，并给出了相应的生成和验证流程。

JSON Web密码应用语法规范
第2部分:数字签名

1 范围

本文件描述了基于JSON数据结构来保护消息内容的数字签名或消息鉴别码的语法规范,并给出了相应的生成和验证流程。

本文件适用于JSON Web密码应用产品的研发与检测,其他使用JSON数据交换格式的安全产品,可参考使用。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中,注日期的引用文件,仅该日期对应的版本适用于本文件;不注日期的引用文件,其最新版本(包括所有的修改单)适用于本文件。

GB/T 15852.2 信息技术 安全技术 消息鉴别码 第2部分:采用专用杂凑函数的机制

GB/T 16263.1 信息技术 ASN.1编码规则 第1部分:基本编码规则(BER)、正则编码规则(CER)和非典型编码规则(DER)规范

GB/T 32905 信息安全技术 SM3密码杂凑算法

GB/T 32918(所有部分) 信息安全技术 SM2椭圆曲线公钥密码算法

GB/T 35276 信息安全技术 SM2密码算法使用规范

GM/T 0125.1 JSON Web密码应用语法规范 第1部分:算法标识

GM/T 0125.4 JSON Web密码应用语法规范 第4部分:密钥

3 术语和定义

下列术语和定义适用于本文件。

3.1

JOSE头部 JOSE header

描述密码运算和参数的JSON对象,由一组参数组成。

3.2

JWS有效载荷 JWS payload

受保护的字节串消息数据,指JWS待签名或消息鉴别的数据。

3.3

JWS签名 JWS signature

对有效载荷和保护头部提供完整性保护的数字签名或消息鉴别码。

3.4

头部参数 header parameter

JOSE头部里面的“参数名称/值”对。

3.5

JWS 保护头部　JWS protected header

受 JWS 数字签名或消息鉴别码完整性保护的头部参数，是一个 JSON 对象。如果是紧凑序列化，保护头部等价于 JOSE 头部。

3.6

JWS 无保护头部　JWS unprotected header

不受 JWS 数字签名或消息鉴别码完整性保护的 JSON 对象。只存在于 JSON 序列化格式中。

3.7

JWS 紧凑序列化　JWS compact serialization

JWS 使用紧凑 URL 安全的字符串形式表示的序列化。

3.8

JWS JSON 序列化　JWS JSON serialization

JWS 使用 JSON 结构字符串表示的序列化。与 JWS 紧凑序列化不同，JSON 序列化可支持无保护头部和多个数字签名或消息鉴别码。

4　缩略语

下列缩略语适用于本文件。

DER：可辨别编码规则(Distinguished Encoding Rules)

JOSE：JSON 签名和加密(JSON Object Signing and Encryption)

JSON：JavaScript 对象标记(JavaScript Object Notation)

JWS：JSON Web 数字签名(JSON Web Signature)

PEM：隐私增强邮件(Privacy Enhanced Mail)

URI：统一资源标识符(Uniform Resource Identifier)

5　符号

下列符号适用于本文件。

ASCII(X)：对数据 X 进行 ASCII 编码

base64url(X)：对数据 X 进行 base64url 编码

UTF8(X)：对数据 X 进行 UTF-8 编码

x ‖ y：x 与 y 的拼接，x、y 是比特串或字节串

6　JSON Web 数字签名(JWS)

6.1　概述

JWS 是使用 JSON 方式来表示对消息数据的数字签名或消息鉴别码的数据结构。

JWS 由 JOSE 头部、JWS 有效载荷、JWS 签名三部分组成。JWS 签名分为两种序列化格式，紧凑序列化和 JSON 序列化。JSON 序列化可包括无保护头部，也可表示多于一个的签名。

6.2　JOSE 头部

6.2.1　通则

JOSE 头部是一个 JSON 对象，JSON 对象的每个参数描述了 JWS 的头部参数，JOSE 头部中的头

部参数名称应唯一。JWS 的 JOSE 头部由 JWS 保护头部和 JWS 无保护头部组成。本文件使用的 DER 编码规则应符合 GB/T 16263.1。

JWS 头部参数名称有三类:已定义参数名称、扩展头部参数名称和自定义头部参数名称。

应用解析和处理已定义的头部参数时应符合 6.2.2 的要求。

6.2.2 已定义的头部参数名称

6.2.2.1 概述

已经定义的头部参数名称具体见表 1。

表 1 JWS 头部参数

参数值	类型	说明	要求
alg	字符串	算法	必选
jku	字符串	密钥资源的 URI	可选
jwk	JSON 对象	JSON Web 密钥	可选
kid	字符串	密钥 ID	可选
x5u	字符串	证书 URL	可选
x5c	数组	证书链	可选
x5t#sm3	字符串	证书 SM3 杂凑值	可选
typ	字符串	整个 JWS 的媒体类型	可选
cty	字符串	JWS 有效载荷媒体类型	可选
crit	数组	应被处理的关键头部参数列表	可选

6.2.2.2 “alg”(算法)头部参数

“alg”头部参数是一个字符串,用于标识 JWS 数字签名或消息鉴别算法。该头部参数必选,取值应符合 GM/T 0125.1。

6.2.2.3 “jku”(JWK Set URL)头部参数

“jku”头部参数是一个 URI 的字符串,它引用一组 JSON 结构的公钥,其中每个公钥对应的私钥用于对 JWS 进行数字签名。公钥应编码为 JWK 集合格式。该头部参数可选,应设置为保护头部参数。

6.2.2.4 “jwk”(JSON Web 密钥)头部参数

“jwk”头部参数是一个 JSON 对象,用于标识对 JWS 进行数字签名的用户,是一个 JSON 对象。该头部参数可选,取值应符合 GM/T 0125.4。

6.2.2.5 “kid”(密钥 ID)头部参数

“kid”头部参数用于标识使用哪个密钥进行 JWS 签名。此头部参数可选,其值以字符串形式表示,区分大小写。当与 JWK 一起使用时,“kid”值用于匹配 JWK 里面的“kid”参数值。

6.2.2.6 “x5u”(证书 URL)头部参数

“x5u”头部参数用于标识对 JWS 进行数字签名的私钥对应的签名证书 URL,其值是一个 URI 形

式的字符串，该资源应提供 PEM 编码形式的证书或证书链的表示，每个证书使用以下分隔符隔开：

-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----

第一张证书应是 JWS 的签名证书。该头部参数可选，应设置为保护头部参数。

6.2.2.7 “x5c”（证书链）头部参数

“x5c”头部参数用于标识 JWS 进行数字签名的签名证书或证书链，其值是一个 JSON 字符串数组。数组中的元素是数字证书 DER 编码的 base64 编码字符串。数组第一个元素应是用于对 JWS 进行数字签名的证书，后面的元素依次是前一个元素颁发证书的 CA 所持有的证书。该头部参数可选。

6.2.2.8 “x5t＃sm3”（证书 SM3 杂凑值）头部参数

“x5t＃sm3” 头部参数是一个字符串，用于标识数字签名证书的 SM3 指纹，可用来匹配密钥。

该值生成过程：使用数字证书的 DER 编码进行 SM3 杂凑后再进行 base64url 编码。SM3 算法的计算方法和步骤应符合 GB/T 32905。该头部参数可选。

6.2.2.9 “typ”（类型）头部参数

“typ”头部参数是一个字符串，用于标识整个 JWS 的媒体类型，该头部参数可选。

6.2.2.10 “cty”（内容类型）头部参数

“cty”头部参数是一个字符串，用于标识 JWS 有效载荷的媒体类型，该头部参数可选。

6.2.2.11 “crit”（关键）头部参数

“crit”头部参数用于标识应解析和处理的头部参数。它是一个 JSON 数组，列出了 JOSE 头部中应处理的头部参数名称，该头部参数可选。

6.2.3 扩展的头部参数名称

本文件可扩展定义新的头部参数。为了防止冲突，禁止新定义的头部参数名称与已定义的头部名称重复。应谨慎引入新的头部参数，避免导致 JWS 之间无法互操作。

6.2.4 自定义头部参数名称

JWS 的应用双方可约定互操作的头部参数名称。与已定义参数名称不同，不同应用的自定义头部参数名称可能会发生冲突，应谨慎使用。

7 JWS 数字签名算法

7.1 概述

JWS 数字签名算法是用于对消息数据进行数字签名的算法。该算法通过 JOSE 头部的“alg”头部参数来标识，标识取值应符合 GM/T 0125.1。

本文件规定了 SM2 数字签名算法的实现要求。JWS 签名的值为数字签名结果。

7.2 SM2 数字签名算法

本节描述了 SM2 数字签名算法的相关要求。

SM2 数字签名算法的签名验证方法应符合 GB/T 32918，指定杂凑算法为 SM3 算法，SM3 算法的

运算方法和步骤应符合 GB/T 32905。

该算法数字签名的输出结果应符合 GB/T 35276,编码规则采用 DER 编码且应符合 GB/T 16263.1。

8 JWS 消息鉴别算法

8.1 概述

JWS 消息鉴别算法是对消息数据提供完整性保护的算法。该算法通过 JOSE 头部的“alg”头部参数来标识,标识取值应符合 GM/T 0125.1。

本文件规定了基于 SM3 的消息鉴别算法的实现要求,JWS 签名的值为消息鉴别算法输出结果。

8.2 基于 SM3 算法的消息鉴别算法

本节描述了基于 SM3 算法的消息鉴别算法的相关要求。

消息鉴别算法的计算过程和结果应符合 GB/T 15852.2,采用 MAC 算法 2,指定杂凑算法为 SM3 算法,SM3 杂凑算法的运算方法和步骤应符合 GB/T 32905。

该算法的输出结果为消息鉴别算法的 HMAC 值,输出长度为 256 比特。

9 JWS 生成和验证

9.1 概述

本章节描述了 JWS 数字签名和消息鉴别码的生成和验证流程。

9.2 生成流程

设消息数据 M 为有效载荷。

计算步骤如下:

a) 计算有效载荷编码为 base64url(M);
b) 根据应用设置的保护头部参数,生成保护头部的 JSON 结构;
c) 如果是 JWS JSON 序列化,根据应用设置的无保护头部参数,生成无保护头部的 JSON 结构;
d) 根据使用的数字签名算法或消息鉴别算法,遵循 GM/T 0125.1,在 JOSE 头部设置“alg”头部参数值;
e) 如果需要设置用户密钥信息,在 JOSE 头部设置相关数据;
f) 保护头部 JSON 结构的字符串使用 UTF-8 编码,计算保护头部编码为 base64url(UTF8(保护头部));
g) 计算待签名或消息鉴别的数据 M=ASCII(保护头部编码‖'.'‖有效载荷编码);
h) 根据 JWS 数字签名或 JWS 消息鉴别算法要求,根据数据 M 计算出 JWS 签名,生成签名编码为 base64url(JWS 签名);
i) 如果使用 JWS JSON 序列化,对每个数字签名或消息鉴别码操作重复此过程[步骤 c)~h)];
j) 输出序列化数据:
   1) 如果是紧凑序列化输出为(保护头部编码‖'.'‖ 有效载荷编码‖'.'‖签名编码);
   2) 如果是 JSON 序列化,输出结果见 12.3。

消息签名和消息鉴别码计算的示例见附录 A。

9.3 验证流程

验证 JWS 是否有效的流程如下。

a) 根据 12.3 解析 JWS 序列化数据。
b) 如果是紧凑序列化，通过'.'分割符解析得到保护头部编码、有效载荷编码和签名编码，如果解析失败，验证不通过。
c) 如果是 JSON 序列化，通过解析 JSON 结构得到一个有效载荷编码、一个或多个保护头部编码、签名编码和无保护头部。
d) 检验有效载荷编码是否 base64url 解码成功，如果检验失败，则验证不通过。
e) 检验保护头部编码是否 base64url 解码成功以及解码后的数据是否符合 UTF-8 编码规则，如果检验失败，则验证不通过。
f) 检验签名编码是否 base64url 解码成功，如果检验失败，则验证不通过。
g) 如果是紧凑序列化，设置 JOSE 头部为保护头部。
h) 如果是 JSON 序列化，设置每个签名者的 JOSE 头部为保护头部和无保护头部的并集。
i) 检验 JOSE 头部是否存在重复头部参数，如果存在重复，验证不通过。
j) 检查 JOSE 头部参数，如果存在“crit”(关键)头部参数值，则应根据参数值定义做相应的处理。
k) 匹配数字签名或消息鉴别码的用户密钥或者标识信息。
l) 检查 JOSE 头部的“alg”头部参数，如果不存在或者标识的算法不支持，则验证不通过。
m) 设置签名原文数据 M＝ASCII(保护头部编码‖'.'‖有效载荷编码)，JWS 签名为签名编码 base64url 解码后的值。根据 JOSE 头部的“alg”头部参数标识出对应的算法，根据算法验证数字签名或消息鉴别码，如果签名或消息鉴别码验证失败，则验证不通过。
n) 如果需要验证多个签名者，重复步骤 e)～m)。
o) 输出验证成功。

10 字符串比较规则

处理 JWS 时，应将已知字符串与 JSON 对象中的头部参数名称和头部参数值进行比较。应严格区分大小写，该比较规则适用于一般情况下对所有 JSON 字符串的比较。当头部参数的定义明确指出要为该头部参数值使用不同的比较规则时，应遵循头部参数的具体定义。

11 密钥身份识别

JWS 的接收者应确定用于数字签名或消息鉴别计算的密钥身份标识。密钥应使用 6.2.2 中描述的头部参数方法标识。

密钥信息可通过头部参数的“jku”“jwk”“kid”“x5u”“x5c”和“x5t＃sm3”来识别，如果该标识需要完整性保护，那么这些头部参数应放到保护头部。

如果应用程序没有使用其他方法或约定来确定所使用的密钥，消息创建则应在头部参数中包含足够的信息来标识所使用的密钥。如果无法识别所使用的密钥，则验证签名或消息鉴别计算失败。

12 序列化

12.1 概述

JWS 有两种序列化表示形式，分别是紧凑序列化和 JSON 序列化。紧凑序列化只支持一个签名对象，头部只包含保护头部；JSON 序列化可支持一个或多个签名对象，头部可包含保护头部和无保护头部。

12.2 紧凑序列化

JWS 紧凑序列化将数字签名或消息鉴别计算结果表示为紧凑的 URL 安全字符串。该字符串表示如下：

base64url(UTF8(JWS 保护头部)) ‖ '.' ‖ base64url(JWS 有效载荷) ‖ '.' ‖ base64url(JWS 签名)

JWS 紧凑格式仅支持一个签名或消息鉴别码。

12.3 JSON 序列化

JSON 序列化将 JWS 签名结果表示为 JSON 对象。

JSON 序列化有两种表现形式。分别是通用 JSON 序列化和扁平化 JSON 序列化。

a) 通用 JSON 序列化：

通用 JSON 序列化是一个 JSON 对象，由 payload 和 signatures 参数组成，见表 2。

signatures 是一个签名对象数组，由 protected，header 和 signature 组成，见表 3。

表 2 通用 JSON 序列化组成参数

参数	类型	内容说明	要求
payload	字符串	JWS 有效载荷使用 base64url 编码后的字符串	必选
signatures	数组	签名对象的数组，数组每个元素代表一个签名对象	必选

表 3 signatures 数组每个元素组成

参数	类型	内容说明	要求
protected	字符串	签名对象的保护头部，JSON 结构的字符串，它是先使用 UTF-8 编码，再进行 base64url 编码后生成的字符串	可选
header	JSON 对象	签名对象的无保护头部，是一个 JSON 对象	可选
signature	字符串	JWS 签名 base64url 编码后的字符串	必选

在序列化结构中“header”“protected”应至少存在一个参数，以确保设置了“alg”头部参数值。“protected”“header”头部参数的并集构成了 JOSE 头部，且并集中的元素名称应不重复。

通用的 JSON 序列化语法如下：

```
{
      "payload":"〈payload contents〉",
      "signatures":[
       {
           "protected":"〈integrity-protected header 1 contents〉",
           "header":〈non-integrity-protected header 1 contents〉,
           "signature":"〈signature 1 contents〉"
       },
           ...
       {
```

```
            "protected":"〈integrity-protected header N contents〉",
            "header":〈non-integrity-protected header N contents〉,
            "signature":"〈signature N contents〉"
        }]
}
```

b) 扁平化 JSON 序列化：

扁平化 JSON 序列化语法基于通用语法，只能表示一个 JWS 签名对象。

语法如下：

```
{
    "payload":"〈payload contents〉",
    "protected":"〈integrity-protected header contents〉",
    "header":〈non-integrity-protected header contents〉,
    "signature":"〈signature contents〉"
}
```

附 录 A
（资料性）
JWS 示例

A.1 综述

本附录中，所有涉及的整数都使用大端格式。

本附录中，明文采用 UTF-8 编码。

本附录中，随机密钥使用 16 进制值表示。

本附录中，签名者的 SM2 私钥不展示出来，在实际使用运用中，SM2 的密钥应由密码模块生成并存放在该密码模块中。

A.2 SM2 算法对有效载荷进行签名示例

A.2.1 概述

该示例展示使用 SM2 数字签名算法签名，使用"x5t＃sm3"来标识证书生成 JWS 的过程和输出结果。

A.2.2 输入参数

输入参数包括以下内容。

a) 有效载荷：message digest

b) 签名算法：“SGD_SM3_SM2”

c) 签名者的 SM2 密钥：

使用如下 JWK 格式来表示：

```
{
    "kty":"EC",
    "crv":"sm2p256v1",
    "use":"sig",
    "x":"TnSVmMedma1KTK20gMTimZGylhJf2JgI8LsYpHosAEg",
    "y":"V0Bn7fBeiPlA66Nzde08dx9culLLjds76HdlaIwvygU"
}
```

“x”和“y”值是对椭圆曲线点坐标分量的字节串进行 base64url 编码后获得的结果。

d) SM2 证书：

SM2 签名密钥的证书 base64 编码如下：

MIIBqzCCAU+gAwIBAgIUfuH5LnZVH2mvK2Qr8MXNJHc7sv0wDAYIKoEcz1UBg3UF
ADAlMQswCQYDVQQGEwJDTjEWMBQGA1UEAwwNU20yX1Jvb3RfVGVzdDAeFw0y
MDAyMjUwMzQwMDdaFw0zNTAyMjUwMzQwMDdaMCAxCzAJBgNVBAYTAkNOMR
EwDwYDVQQDDAhTTTIgU2lnbjBZMBMGByqGSM49AgEGCCqBHM9VAYItA0IABE50l
ZjHnZmtSkyttIDE4pmRspYSX9iYCPC7GKR6LABIV0Bn7fBeiPlA66Nzde08dx9culLLjds76
HdlaIwvygWjYDBeMB8GA1UdIwQYMBaAFKoL1nBIYnHh5BMWrD84Tx0tJ3r5MB0GA1
UdDgQWBBQcPQghGArysGLq7AXU6WE6VyqG+DAMBgNVHRMBAf8EAjAAMA4GA1
UdDwEB/wQEAwIGwDAMBggqgRzPVQGDdQUAA0gAMEUCIQC3Joxo7p5nfP4oa6Gsl4

BPlWu9sPD8Lv/xxHtHrikfnAIgVb55YLDbzIMkf1TCaGIewUMDLeIsGVHHBudnXOWrlzw=

A.2.3 计算过程

计算过程包括以下内容。

a) 计算 base64url(有效载荷),有效载荷编码结果为:bWVzc2FnZSBkaWdlc3Q

b) 根据 SM2 证书,使用 SM3 算法计算证书 DER 编码的杂凑值,对杂凑值进行 base64url 编码,其结果为:pUrOSKoNG_tEzuVJeVxZQPTyAx13qPSuYT_4-esJL2A

c) 根据签名算法“SGD_SM3_SM2”生成保护头部的 JSON 结构

```
{
    "alg":"SGD_SM3_SM2",
    "x5t#sm3":"pUrOSKoNG_tEzuVJeVxZQPTyAx13qPSuYT_4-esJL2A"
}
```

JSON 结构的字符串使用 UTF-8 编码,进行 base64url 编码,生成保护头部编码=
eyJhbGciOiJTR0RfU00zX1NNMiIsIng1dCNzbTMiOiJwVXJPU0tvTkdfdEV6dVZKZVZ4WlFQVHlBeDEzcVBTdVlUXzQtZXNKTDJBIn0

生成签名的输入参数 M=ASCII(保护头部编码‖'.'‖有效载荷编码),对应结果为:eyJhbGciOiJTR0RfU00zX1NNMiIsIng1dCNzbTMiOiJwVXJPU0tvTkdfdEV6dVZKZVZ4WlFQVHlBeDEzcVBTdVlUXzQtZXNKTDJBIn0.bWVzc2FnZSBkaWdlc3Q

d) 把 M 作为待签名对象,按照 SM2 数字签名算法要求,使用 SM2 签名密钥计算签名值,对签名值进行 base64url 编码,最终签名结果的编码为:MEQCIEs0XaIIreO2LQmwQl9OVv2yRlUXsTV3AxqKfmT619tGAiBi7XyPsRPmv0U5asV63d950fWq2BzwaSK1xd85hAAnFg

A.2.4 输出结果

根据上面计算的保护头部编码,有效载荷编码,签名结果编码,输出紧凑序列化和 JSON 序列化结果如下。

a) 紧凑序列化:

eyJhbGciOiJTR0RfU00zX1NNMiIsIng1dCNzbTMiOiJwVXJPU0tvTkdfdEV6dVZKZVZ4WlFQVHlBeDEzcVBTdVlUXzQtZXNKTDJBIn0.bWVzc2FnZSBkaWdlc3Q.MEQCIEs0XaIIreO2LQmwQl9OVv2yRlUXsTV3AxqKfmT619tGAiBi7XyPsRPmv0U5asV63d950fWq2BzwaSK1xd85hAAnFg

b) JSON 序列化:

```
{
    "payload":"bWVzc2FnZSBkaWdlc3Q",
    "protected":"eyJhbGciOiJTR0RfU00zX1NNMiIsIng1dCNzbTMiOiJwVXJPU0tvTkdfdEV6dVZKZVZ4WlFQVHlBeDEzcVBTdVlUXzQtZXNKTDJBIn0",
    "signature":"MEQCIEs0XaIIreO2LQmwQl9OVv2yRlUXsTV3AxqKfmT619tGAiBi7XyPsRPmv0U5asV63d950fWq2BzwaSK1xd85hAAnFg"
}
```

A.3 SM2 算法对有效载荷生成消息鉴别码示例

A.3.1 概述

该示例展示使用基于 SM3 的消息鉴别算法来生成 JWS 的过程和输出结果。

A.3.2 输入参数

输入参数包括以下内容。

a) 有效载荷：message hmac

b) 算法："SGD_SM3_HMAC"

c) 密钥 Key：16 进制表示为
313233343536373831323334353637383132333435363738313233343536373831323334353637383132333435363738

A.3.3 计算过程

计算过程包括以下内容。

a) 计算 base64url(有效载荷)，有效载荷编码结果为：bWVzc2FnZSBobWFj

b) 根据算法"SGD_SM3_HMAC"生成保护头部的 JSON 结构
{"alg":"SGD_SM3_HMAC"}
JSON 结构的字符串使用 UTF-8 编码，对其进行 base64url 编码，生成的保护头部编码为：
eyJhbGciOiJTR0RfU00zX0hNQUMifQ

c) 生成 HMAC 的输入参数 M＝ASCII(保护头部编码‖'.'‖有效载荷编码)，对应结果为：
eyJhbGciOiJTR0RfU00zX0hNQUMifQ.bWVzc2FnZSBobWFj

d) 把 M 和密钥 Key 作为 HMAC 算法参数，按照基于 SM3 算法的消息鉴别算法要求计算出 HMAC 值，对值进行 base64url 编码，最终结果编码为：
yCN-3KJb5RIW9pBunTtHFPQKZmzRnMy2bxGFaBuUuyU

A.3.4 输出结果

根据上面计算的保护头部编码、有效载荷编码、HMAC 结果编码，输出紧凑序列化和 JSON 序列化结果如下。

a) 紧凑序列化：
eyJhbGciOiJTR0RfU00zX0hNQUMifQ.bWVzc2FnZSBobWFj.yCN-3KJb5RIW9pBunTtHFPQKZmzRnMy2bxGFaBuUuyU

b) JSON 序列化：

```
{
    "payload":"bWVzc2FnZSBobWFj",
    "protected":"eyJhbGciOiJTR0RfU00zX0hNQUMifQ",
    "signature":"yCN-3KJb5RIW9pBunTtHFPQKZmzRnMy2bxGFaBuUuyU"
}
```

A.4 SM2 算法对有效载荷进行多人签名示例

A.4.1 概述

该示例展示两个签名者各自独立使用 SM2 数字签名算法对同一个消息签名，一个签名者使用"x5t＃sm3"头部参数来标识证书，另外一个签名者使用"x5c"头部参数来标识证书，在一项 JWS 输出中同时包含了这两项签名，两个签名者不分先后顺序，属于并行关系，输入参数、计算过程、输出结果如下。

A.4.2 输入参数

输入参数包括以下内容。

a) 有效载荷：message digest

b) 签名算法："SGD_SM3_SM2"

c) 第一个签名者的 SM2 密钥：

使用如下 JWK 格式来表示：


```
{
    "kty":"EC",
    "crv":"sm2p256v1",
    "use":"sig",
    "x":"TnSVmMedma1KTK20gMTimZGylhJf2JgI8LsYpHosAEg",
    "y":"V0Bn7fBeiPlA66Nzde08dx9culLLjds76HdlaIwvygU"
}
```


"x"和"y"值是对椭圆曲线点坐标分量的字节串进行 base64url 编码后获得的结果。

d) 第一个签名者的 SM2 证书：

SM2 证书 base64 编码如下：

MIIBqzCCAU+gAwIBAgIUfuH5LnZVH2mvK2Qr8MXNJHc7sv0wDAYIKoEcz1UBg3UF
ADAlMQswCQYDVQQGEwJDTjEWMBQGA1UEAwwNU20yX1Jvb3RfVGVzdDAeFw0y
MDAyMjUwMzQwMDdaFw0zNTAyMjUwMzQwMDdaMCAxCzAJBgNVBAYTAkNOMR
EwDwYDVQQDDAhTTTIgU2lnbjBZMBMGByqGSM49AgEGCCqBHM9VAYItA0IABE50l
ZjHnZmtSkyttIDE4pmRspYSX9iYCPC7GKR6LABIV0Bn7fBeiPlA66Nzde08dx9culLLjds76
HdlaIwvygWjYDBeMB8GA1UdIwQYMBaAFKoL1nBIYnHh5BMWrD84Tx0tJ3r5MB0GA1
UdDgQWBBQcPQghGArysGLq7AXU6WE6VyqG+DAMBgNVHRMBAf8EAjAAMA4GA1
UdDwEB/wQEAwIGwDAMBggqgRzPVQGDdQUAA0gAMEUCIQC3Joxo7p5nfP4oa6Gsl4BPlWu
9sPD8Lv/xxHtHrikfnAIgVb55YLDbzIMkf1TCaGIewUMDLeIsGVHHBudnXOWrlzw=

e) 第二个签名者的 SM2 密钥：

使用 JWK 格式表示如下：


```
{
    "kty":"EC",
    "crv":"sm2p256v1",
    "use":"sig",
    "x":"VS6TxmGXIPad18vqf9XTqYQz5s4Dx4TN-obVuEydnFg",
    "y":"UHDXtrT7kP1J8B1v3vSXw8-o-mHcy7TiDJe3UL07wOw"
}
```


"x"和"y"值是对椭圆曲线点坐标的字节串进行 base64url 编码后获得的结果。

f) 第二个签名者的 SM2 证书：

SM2 的证书 base64 编码如下：

MIIBrDCCAVCgAwIBAgIUJKLZ35ihzJhAB/kyPraSSRhLeVAwDAYIKoEcz1UBg3UFAD
AlMQswCQYDVQQGEwJDTjEWMBQGA1UEAwwNU20yX1Jvb3RfVGVzdDAeFw0yMD
AzMDIxMjA3NDdaFw0zNTAzMDIxMjA3NDdaMCExCzAJBgNVBAYTAkNOMRIwEAYD
VQQDDAlTTTIgU2lnbjIwWTATBgcqhkjOPQIBBggqgRzPVQGCLQNCAARVLpPGYZcg
9p3Xy+p/1dOphDPmzgPHhM36htW4TJ2cWFBw17a0+5D9SfAdb970l8PPqPph3Mu04gyX
t1C9O8Dso2AwXjAfBgNVHSMEGDAWgBSqC9ZwSGJx4eQTFqw/OE8dLSd6+TAdBgNV
HQ4EFgQU+uhsyI0m+sjlL9yiQ9W5D/gPbEkwDAYDVR0TAQH/BAIwADAOBgNVHQ
8BAf8EBAMCBsAwDAYIKoEcz1UBg3UFAANIADBFAiAZrGkjYgLD8X+P++D0Ufr/K

Eo7irKD+sY6nERXGKM9JAIhAKjGkTc5suOhBKYZ//iXuSvLMUIzQQ1SneiFzfL97Oz1

A.4.3 计算过程

计算过程包括以下内容。

a) 计算 base64url(有效载荷),有效载荷编码结果为:bWVzc2FnZSBkaWdlc3Q

b) 计算第一个签名:

1) 根据 SM2 证书,使用 SM3 算法计算证书 DER 编码的杂凑值,对杂凑值进行 base64url 编码,其结果为:pUrOSKoNG_tEzuVJeVxZQPTyAx13qPSuYT_4-esJL2A

2) 根据签名算法“SGD_SM3_SM2”生成保护头部的 JSON 结构

```
{
    "alg":"SGD_SM3_SM2",
    "x5t#sm3":"pUrOSKoNG_tEzuVJeVxZQPTyAx13qPSuYT_4-esJL2A"
}
```

JSON 结构的紧凑格式字符串使用 UTF-8 编码,对其进行 base64url 编码,生成的保护头部编码为:

eyJhbGciOiJTR0RfU00zX1NNMiIsIng1dCNzbTMiOiJwVXJPU0tvTkdfdEV6dVZKZVZ4WlFQVHlBeDEzcVBTdVlUXzQtZXNKTDJBIn0

3) 生成签名的输入参数 M=ASCII(保护头部编码||'.'||有效载荷编码),对应结果为:

eyJhbGciOiJTR0RfU00zX1NNMiIsIng1dCNzbTMiOiJwVXJPU0tvTkdfdEV6dVZKZVZ4WlFQVHlBeDEzcVBTdVlUXzQtZXNKTDJBIn0.bWVzc2FnZSBkaWdlc3Q

4) 把 M 和 SM2 密钥作为签名参数,按照 SM2 数字签名算法要求,计算出签名值,将签名值进行 base64url 编码,最终签名结果编码为:

MEYCIQDpamLPgihKxRUuRaqZy-nOaarpBzeV0LbvL8EXC64g2wIhAL5Lb9CbsrrYc88PkFzMMxqn0iFZU0HXnYyGGxU3W32-

c) 计算第二个签名:

1) 根据签名算法“SGD_SM3_SM2”和 SM2 证书生成保护头部的 JSON 结构,其内容为:

```
{
    "alg":"SGD_SM3_SM2",
    "x5c":["MIIBrDCCAVCgAwIBAgIUJKLZ35ihzJhAB/kyPraSSRhLeVAwDAYIKoEcz1UBg3UFADAlMQswCQYDVQQGEwJDTjEWMBQGA1UEAwwNU20yX1Jvb3RfVGVzdDAeFw0yMDAzMDIxMjA3NDdaFw0zNTAzMDIxMjA3NDdaMCExCzAJBgNVBAYTAkNOMRIwEAYDVQQDDAlTTTIgU2lnbjIwWTATBgcqhkjOPQIBBggqgRzPVQGCLQNCAARVLpPGYZcg9p3Xy+p/1dOphDPmzgPHhM36htW4TJ2cWFBw17a0+5D9SfAdb970l8PPqPph3Mu04gyXt1C9O8Dso2AwXjAfBgNVHSMEGDAWgBSqC9ZwSGJx4eQTFqw/OE8dLSd6+TAdBgNVHQ4EFgQU+uhsyI0m+sjlL9yiQ9W5D/gPbEkwDAYDVR0TAQH/BAIwADAOBgNVHQ8BAf8EBAMCBsAwDAYIKoEcz1UBg3UFAANIADBFAiAZrGkjYgLD8X+P++D0Ufr/KEo7irKD+sY6nERXGKM9JAIhAKjGkTc5suOhBKYZ//iXuSvLMUIzQQ1SneiFzfL97Oz1"]
}
```

JSON 结构的字符串使用 UTF-8 编码,对其进行 base64url 编码,生成的保护头部编码为:

eyJhbGciOiJTR0RfU00zX1NNMiIsIng1YyI6WyJNSUlCckRDQ0FWQ2dBd0lCQWdJVUpLTFozNWloekpoQUIva3lQcmFTU1JoTGVWQXdEQVlJS29FY3oxVUJnM1VGQURBbE1Rc3dDUVlEVlFRR0V3SkRUakVXTUJRR0ExVUVBd3dOVTIweVgxSnZiM1J

mVkdWemREQWVGdzB5TURBek1ESXhNakEzTkRkYUZ3MHpOVEF6TURJeE1qQ
TNORGRhTUNFeEN6QUpCZ05WQkFZVEFrTk9NUkl3RUFZRFZRUUREQWxUVF
RJZ1UybG5iakl3V1RBVEJnY3Foa2pPUFFJQkJnZ3FnUnpQVlFHQ0xRTkNBQVJWT
HBQR1laY2c5cDNYeStwLzFkT3BoRFBtemdQSGhNMzZodFc0VEoyY1dGQncxN2Ew
KzVEOVNmQWRiOTcwbDhQUHFQcGgzTXUwNGd5WHQxQzlPOERzbzJBd1hqQW
ZCZ05WSFNNRUdEQVdnQlNxQzlad1NHSng0ZVFURnF3L09FOGRMU2Q2K1RBZE
JnTlZIUTRFRmdRVSt1aHN5STBtK3NqbEw5eWlROVc1RC9nUGJFa3dEQVlEVlIwV
EFRSC9CQUl3QURBT0JnTlZIUThCQWY4RUJBTUNCc0F3REFZSUtvRWN6MVVC
ZzNVRkFBTklBREJGQWlBWnJHa2pZZ0xEOFgrUCsrRDBVZnIvS0VvN2lyS0Qrc1k2b
kVSWEdLTTlKQUloQUtqR2tUYzVzdU9oQktZWi8vaVh1U3ZMTVVJelFRMVNuZW
lGemZMOTdPejEiXX0

2) 生成签名的输入参数 M= ASCII(保护头部编码‖'.'‖有效载荷编码),对应结果为:
eyJhbGciOiJTR0RfU00zX1NNMiIsIng1YyI6WyJNSUlCckRDQ0FWQ2dBd0lCQWdJV
UpLTFozNWloekpoQUIva3lQcmFTU1JoTGVWQXdEQVlJS29FY3oxVUJnM1VGQU
RBbE1Rc3dDUVlEVlFRR0V3SkRUakVXTUJRR0ExVUVBd3dOVTIweVgxSnZiM1J
mVkdWemREQWVGdzB5TURBek1ESXhNakEzTkRkYUZ3MHpOVEF6TURJeE1qQ
TNORGRhTUNFeEN6QUpCZ05WQkFZVEFrTk9NUkl3RUFZRFZRUUREQWxUVF
RJZ1UybG5iakl3V1RBVEJnY3Foa2pPUFFJQkJnZ3FnUnpQVlFHQ0xRTkNBQVJWT
HBQR1laY2c5cDNYeStwLzFkT3BoRFBtemdQSGhNMzZodFc0VEoyY1dGQncxN2Ew
KzVEOVNmQWRiOTcwbDhQUHFQcGgzTXUwNGd5WHQxQzlPOERzbzJBd1hqQW
ZCZ05WSFNNRUdEQVdnQlNxQzlad1NHSng0ZVFURnF3L09FOGRMU2Q2K1RBZE
JnTlZIUTRFRmdRVSt1aHN5STBtK3NqbEw5eWlROVc1RC9nUGJFa3dEQVlEVlIwV
EFRSC9CQUl3QURBT0JnTlZIUThCQWY4RUJBTUNCc0F3REFZSUtvRWN6MVVC
ZzNVRkFBTklBREJGQWlBWnJHa2pZZ0xEOFgrUCsrRDBVZnIvS0VvN2lyS0Qrc1k2b
kVSWEdLTTlKQUloQUtqR2tUYzVzdU9oQktZWi8vaVh1U3ZMTVVJelFRMVNuZW
lGemZMOTdPejEiXX0.bWVzc2FnZSBkaWdlc3Q

3) 把 M 和 SM2 密钥作为签名参数,按照 SM2 数字签名算法要求,计算出签名值,对签名值进行 base64url 编码,最终签名结果编码为:
MEYCIQCX8PtbD8hd0pcFG5g0vzLaMAi3YTCx5NgPTtAr1ZExuQIhALJrj8fgDSrR5b
O-fDtZHA0q7EcYD1j9PXCmp0mbtiHX

A.4.4 输出结果

根据上面计算的保护头部编码、有效载荷编码、签名结果编码,输出 JSON 序列化结果如下(多人签名不支持紧凑序列化)。

JSON 序列化:

{"payload":"bWVzc2FnZSBkaWdlc3Q",

"signatures":[{"protected":"eyJhbGciOiJTR0RfU00zX1NNMiIsIng1dCNzbTMiOiJwVXJP
U0tvTkdfdEV6dVZKZVZ4WlFQVHlBeDEzcVBTdVlUXzQtZXNKTDJBIn0","signature":"ME
YCIQDpamLPgihKxRUuRaqZy-nOaarpBzeV0LbvL8EXC64g2wIhAL5Lb9CbsrrYc88PkFzMMxq
n0iFZU0HXnYyGGxU3W32-"},{"protected":"eyJhbGciOiJTR0RfU00zX1NNMiIsIng1YyI6Wy
JNSUlCckRDQ0FWQ2dBd0lCQWdJVUpLTFozNWloekpoQUIva3lQcmFTU1JoTGVWQXdEQV
lJS29FY3oxVUJnM1VGQURBbE1Rc3dDUVlEVlFRR0V3SkRUakVXTUJRR0ExVUVBd3dOV
TIweVgxSnZiM1JmVkdWemREQWVGdzB5TURBek1ESXhNakEzTkRkYUZ3MHpOVEF6TU

RJeE1qQTNORGRhTUNFeEN6QUpCZ05WQkFZVEFrTk9NUkl3RUFZRFZRUUREQWxUVF
RJZ1UybG5iakl3V1RBVEJnY3Foa2pPUFFJQkJnZ3FnUnpQVlFHQ0xRTkNBQVJWTHBQR1la
Y2c5cDNYeStwLzFkT3BoRFBtemdQSGhNMzZodFc0VEoyY1dGQncxN2EwKzVEOVNmQWRi
OTcwbDhQUHFQcGgzTXUwNGd5WHQxQzlPOERzbzJBd1hqQWZCZ05WSFNNRUdEQVdnQ
lNxQzlad1NHSng0ZVFURnF3L09FOGRMU2Q2K1RBZEJnTlZIUTRFRmdRVSt1aHN5STBtK
3NqbEw5eWlROVc1RC9nUGJFa3dEQVlEVlIwVEFRSC9CQUl3QURBT0JnTlZIUThCQWY4R
UJBTUNCc0F3REFZSUtvRWN6MVVCZzNVRkFBTklBREJGQWlBWnJHa2pZZ0xEOFgrUCsr
RDBVZnIvS0VvN2lyS0Qrc1k2bkVSWEdLTTlKQUloQUtqR2tUYzVzdU9oQktZWi8vaVh1U3Z
MTVVJelFRMVNuZWlGemZMOTdPejEiXX0"，"signature"："MEYCIQCX8PtbD8hd0pcFG5g0vzLaMA
i3YTCx5NgPTtAr1ZExuQIhALJrj8fgDSrR5bO-fDtZHA0q7EcYD1j9PXCmp0mbtiHX"}]}