ICS 35.040
CCS L 80

中华人民共和国密码行业标准

GM/T 0087—2020

浏览器密码应用接口规范

Browser cryptography API specification

2020-12-28 发布　　2021-07-01 实施

国家密码管理局　　发布

目　　次

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。

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：北京海泰方圆科技股份有限公司、无锡江南信息安全工程技术中心、格尔软件股份有限公司、成都卫士通信息产业股份有限公司、吉大正元信息技术股份有限公司。

本文件主要起草人：柳增寿、蒋红宇、徐明翼、郑强、罗俊、赵丽丽。

浏览器密码应用接口规范

1 范围

本文件定义了浏览器执行网页中的密码操作的 JavaScript API，包括加密、解密、杂凑、签名、签名验证和随机数生成等操作。

本文件定义的 API 适用于浏览器中用户或服务的认证、文档或代码的签名、通信的机密性与完整性保证等。

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 32918（所有部分） 信息安全技术 SM2 椭圆曲线公钥密码算法

GB/T 32905 信息安全技术 SM3 密码杂凑算法

GB/T 32907 信息安全技术 SM4 分组密码算法

3 术语、定义和缩略语

3.1 术语和定义

下列术语和定义适用于本文件。

3.1.1

字典

一种使用键-值对作为元素的集合。

3.1.2

承诺

一种 JavaScript 范式，承诺（promise）代表一个任务结果。通过本范式可以实现浏览器脚本程序的异步功能。

3.2 缩略语

下列缩略语适用于本文件。

IDL 接口描述语言（Interface Description Language）

4 概述

本文件用于为网络应用中浏览器 JavaScript 脚本提供密码操作能力。网络应用可以让用户利用浏览器内置密码能力在浏览器端来保护其身份数据和隐私数据。直接使用 JavaScript 实现密码功能的方式会导致安全缺陷和性能问题。因此有必要在浏览器上原生实现密码功能，并向 JavaScript 程序提供

密码支撑。

安全接口 Crypto 提供了 WCAPI 的通用加密功能的接口。该接口定义一个 crypto 对象开放给浏览器，在 JavaScript 中可以全局访问。crypto 对象包含生成随机数的方法和一个 subtle 对象。subtle 对象实现了 SubtleCrypto 接口。SubtleCrypto 接口提供了对若干具体的密码计算的访问接口。

用户 JavaScript 程序可以使用的密码资源的层次关系见图 1。

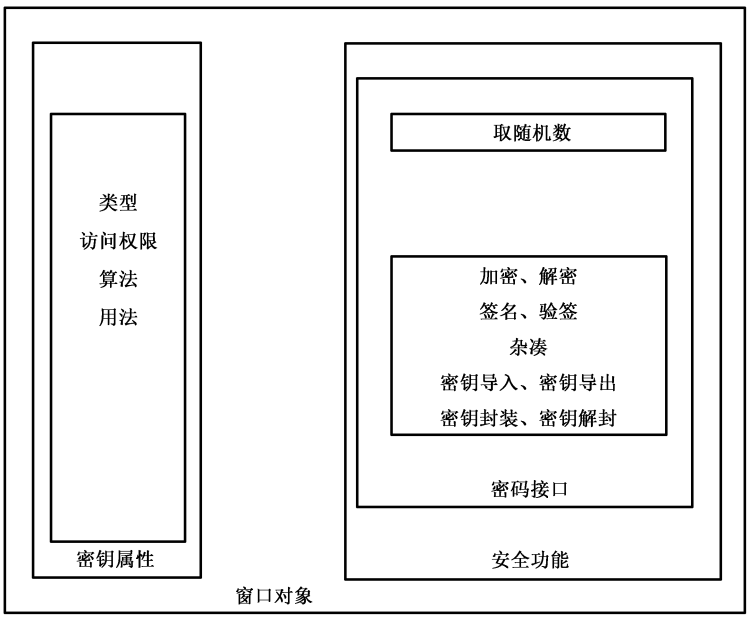

图 1　密码资源的层次关系

5　数据结构

5.1　大整数

IDL：

```
typedef Uint8Array BigInteger;
```

大整数 BigInteger 定义为 Uint8Array 类型，保存任意长度的无符号整数，高字节在前。从 API 读取的数值应有最小的类型化数组的长度(即除了 0 之外，至多 7 个前导 0 比特)。API 应接受任意数目的前导 0 比特。

5.2　密钥对字典

IDL：

```
dictionary CryptoKeyPair
{
  CryptoKey publicKey;
  CryptoKey privateKey;
};
```

CryptoKeyPair 密钥对字典表示一对非对称密钥对，该密钥对由私钥和公钥两部分组成。

5.3　JsonWebKey 字典

IDL：

```
dictionaryRsaOtherPrimesInfo
{
  DOMString r;
  DOMString d;
  DOMString t;
};
dictionary JsonWebKey
{
  DOMString kty;
  DOMString use;
  sequence〈DOMString〉 key_ops;
  DOMString alg;
  boolean ext;
  DOMString crv;
  DOMString x;
  DOMString y;
  DOMString d;
  DOMString n;
  DOMString e;
  DOMString p;
  DOMStrinβg q;
  DOMString dp;
  DOMString dq;
  DOMString qi;
  sequence〈RsaOtherPrimesInfo〉 oth;
  DOMString k;
};
```

JsonWebKey 字典提供了一种表示和交换表示为 JSON 网页密钥机构的密钥的方法，同时允许在浏览器密码 API 应用中进行本地的高效调用。

5.4 算法字典 Algorithm

算法对象是一个网页 IDL 字典对象，它用于为特定操作来指定算法和附加参数。

IDL：

```
typedef (object or DOMString) AlgorithmIdentifier;
typedef AlgorithmIdentifier HashAlgorithmIdentifier;
dictionary Algorithm {required DOMString name; };
```

其中 name 为待用的已注册算法的名字。

5.5 密码接口 Crypto

IDL：

```
[NoInterfaceObject]
interface GlobalCrypto
{
  readonly attribute Crypto crypto;
```

```
};
Window implements GlobalCrypto;
WorkerGlobalScope implements GlobalCrypto;
[Exposed=(Window,Worker)]
interface Crypto
{
  readonly attribute SubtleCrypto subtle;
  ArrayBufferView getRandomValues(ArrayBufferView array);
};
```

密码接口提供了通用密码功能的接口，其中包含了一个使用真随机值作为种子的密码学强伪随机数生成器。

其中，getRandomValues 方法用于生成随机数。该方法应遵循：

a) 若 array 不为整数类型（如 Int8Array，Uint8Array，Int16Array，Uint16Array，Int32Array，或 Uint32Array），则抛出 TypeMismatchError 异常并结束计算。

b) 若数组的 byteLength 大于 65536，则抛出 QuotaExceededError 异常并结束算法。

c) 用适当类型的密码学随机数对数组的所有元素覆盖。

d) 返回 array。

subtle 属性提供了 SubtleCrypto 接口的一个实例，该接口提供了底层的密码功能和算法。

5.6 密钥算法 KeyAlgorithm

KeyAlgorithm 字典表示与一个给定的 CryptoKey 对象的内容相关信息。

IDL：

```
dictionary KeyAlgorithm {required DOMString name };
```

KeyAlgorithm 字典用于将 CryptoKey 的固定公开属性进行归档。实际字典的类型并不向应用进行开放。

其中，字段成员 name 为用于生成 CryptoKey 对象的算法的名字。

5.7 密钥接口 CryptoKey

IDL：

```
enum KeyType { "public", "private", "secret" };
enum KeyUsage
{
"encrypt", "decrypt", "sign", "verify", "deriveKey",
"deriveBits", "wrapKey", "unwrapKey"
};
[Exposed=(Window,Worker)]
interface CryptoKey
{
  readonly attribute KeyTypetype;
  readonly attribute boolean extractable;
  readonly attribute object algorithm;
  readonly attribute object usages;
};
```

其中，

KeyType 为密钥的类型。可识别的密钥类型值为“public”“private”和“secret”。非透明密钥包括对称算法密钥用“secret”表示，非对称算法密钥由公和私钥组成，分别由“public”或“private”表示。

KeyUsage 为密钥的操作类型。可识别的密钥用法包括“encrypt”“decrypt”“sign”“verify”“deriveKey”“deriveBits”“wrapKey”和“unwrapKey”。

每个 CryptoKey 对象有一个内部组成集合，用于存储密钥相关的信息。这些组成部分不作为标准的一部分而暴露。他们表示具体实现本规范时的内部状态。内部组成通过双方括号[[ ]]内的名字进行表述。

CryptoKey 对象内部包括[[type]]，[[extractable]]，[[algorithm]]，[[algorithm_cached]]，[[usages]]，[[usages_cached]]和[[handle]]。

[[algorithm]]内部组成应为、或者继承自 KeyAlgorithm。[[usages]]内部组成的内容应为 Sequence〈KeyUsage〉类型。

Type 反映了[[type]]内部组成，它包含了依赖密钥的类型。

extractable 反映了[[extractable]]内部组成，它指明了密钥是否可以导出至应用。

algorithm 返回与[[algorithm]]内部组成相关的缓冲脚本对象。

usages 返回与[[usages]]内部组成相关的缓冲脚本对象，指明该密钥允许哪些密码操作。

6 密码接口

6.1 接口定义

IDL：

```
enum KeyFormat { “raw”, “spki”, “pkcs8”, “jwk” };
[Exposed=(Window,Worker)]
interface SubtleCrypto
{
  Promise〈any〉 encrypt(AlgorithmIdentifier algorithm, CryptoKey key, BufferSource data);
  Promise〈any〉 decrypt(AlgorithmIdentifier algorithm, CryptoKey key, BufferSource data);
  Promise〈any〉sign(AlgorithmIdentifier algorithm, CryptoKey key, BufferSource data);
  Promise〈any〉 verify(AlgorithmIdentifier algorithm, CryptoKey key, BufferSource signature,
  BufferSource data);
  Promise〈any〉 digest(AlgorithmIdentifier algorithm, BufferSource data);
  Promise〈any〉 generateKey(AlgorithmIdentifier algorithm, boolean extractable, sequence
  〈KeyUsage〉 keyUsages );
  Promise〈any〉 deriveKey(AlgorithmIdentifier algorithm, CryptoKey baseKey, AlgorithmIdentifier derivedKeyType, boolean extractable, sequence〈KeyUsage〉 keyUsages );
  Promise〈any〉 deriveBits(AlgorithmIdentifier algorithm, CryptoKey baseKey,
  unsigned long length);
  Promise〈any〉 importKey(KeyFormatformat, (BufferSource or JsonWebKey) keyData, AlgorithmIdentifier algorithm, boolean extractable, sequence〈KeyUsage〉 keyUsages );
  Promise〈any〉 exportKey(KeyFormat format, CryptoKey key);
  Promise〈any〉 wrapKey(KeyFormat format, CryptoKey key, CryptoKey wrappingKey, AlgorithmIdentifier wrapAlgorithm);
  Promise〈any〉 unwrapKey(KeyFormat format, BufferSource wrappedKey, CryptoKey unwrappingKey, AlgorithmIdentifier unwrapAlgorithm, AlgorithmIdentifier unwrappedKeyAlgorithm, boolean extractable, sequence〈KeyUsage〉 keyUsages );
```

};

其中，KeyFormat 指定密钥串行化的格式。可识别的密钥格式取值为：

“raw”

非格式化字节序列。用于秘密密钥。

“pkcs8”

符合 RFC5280 格式的私钥信息格式的 DER 编码。

“jwk”

密钥作为 JsonWebKey 对象并为 JavaScript 对象。

6.2 加密方法

encrypt 加密方法返回一个新的承诺对象，该对象使用指定的 AlgorithmIdentifier 和所提供的 CryptoKey 来加密数据。它应执行如下步骤：

a) 将算法和密钥分别作为 algorithm 和 key 参数传递至加密方法；
b) 将数据作为 data 参数的克隆数据结果传递至加密算法；
c) 将 normalizedAlgorithm 作为正规化算法的结果，将 alg 设置为 algorithm，将 op 设置为“encrypt”；
d) 若出错，返回带有 normalizedAlgorithm 的承诺拒绝；
e) 设置 promise 为新的 Promise；
f) 返回 promise 并异步完成余下的步骤；
g) 若余下的步骤或者调用的过程抛出异常或者拒绝承诺，则结束算法；
h) 若 normalizedAlgorithm 的 name 成员不同于密钥的[[algorithm]]内部组织的 name 属性，抛出 InvalidAccessError；
i) 若密钥的[[usages]]内部组织不包含“encrypt”条目，抛出 InvalidAccessError；
j) 将 ciphertext 设置为加密操作的结果，该操作过程指定了 normalizedAlgorithm 加密操作、密钥和明文；
k) 用 ciphertext 作为 Promise 对象返回。

6.3 解密方法

decrypt 解密方法返回一个新的承诺对象，该对象使用 AlgorithmIdentifier 和提供的 CryptoKey 对密文进行解密。它应执行下列步骤：

a) 将算法和密钥分别作为 algorithm 和 key 参数传递至解密方法；
b) 将数据作为 data 参数的克隆数据结果传递至解密算法；
c) 将 normalizedAlgorithm 作为正规化算法的结果，将 alg 设置为 algorithm，将 op 设置为“decrypt”；
d) 若出错，返回带有 normalizedAlgorithm 的承诺拒绝；
e) 设置 promise 为新的 Promise；
f) 返回 promise 并异步完成余下的步骤；
g) 若余下的步骤或者调用的过程抛出异常或者拒绝承诺，则结束算法；
h) 若 normalizedAlgorithm 的 name 成员不同于密钥的[[algorithm]]内部组织的 name 属性，抛出 InvalidAccessError；
i) 若密钥的[[usages]]内部组织不包含“decrypt”条目，抛出 InvalidAccessError；
j) 将 plaintext 设置为解密操作的结果，该操作过程指定了特定密钥、算法和密文下的 normalizedAlgorithm 解密操作；
k) 用 plaintext 作为 Promise 对象返回。

6.4 签名方法

sign 签名方法返回一个新的承诺对象，该对象使用 AlgorithmIdentifier 和提供的 CryptoKey 对数据进行签名。它应遵循下列步骤：

a) 将算法和密钥分别作为 algorithm 和 key 参数传递至签名方法；
b) 将数据作为 data 参数的克隆数据结果传递至签名算法；
c) 将 normalizedAlgorithm 作为正规化算法的结果，将 alg 设置为 algorithm，将 op 设置为“sign”；
d) 若出错，返回带有 normalizedAlgorithm 的承诺拒绝；
e) 设置 promise 为新的 Promise；
f) 返回 promise 并异步完成余下的步骤；
g) 若余下的步骤或者调用的过程抛出异常或者拒绝承诺，则结束算法；
h) 若 normalizedAlgorithm 的 name 成员不同于密钥的[[algorithm]]内部组织的 name 属性，抛出 InvalidAccessError；
i) 若密钥的[[usages]]内部组织不包含“sign”条目，抛出 InvalidAccessError；
j) 将 plaintext 设置为解密操作的结果，该操作过程指定了特定密钥、算法和消息数据下的 normalizedAlgorithm 签名操作；
k) 将 Promise 对象返回。

6.5 验证签名方法

verify 验证签名方法返回一个新的承诺对象，该对象使用 AlgorithmIdentifier 和提供的 CryptoKey 对数据进行验签。它应遵循下列步骤：

a) 将算法和密钥分别作为 algorithm 和 key 参数传递至验签方法；
b) 将签名数据作为签名参数的 data 克隆数据结果传递至验签算法；
c) 将 normalizedAlgorithm 作为正规化算法的结果，将 alg 设置为 algorithm，将 op 设置为“verify”；
d) 若出错，返回带有 normalizedAlgorithm 的承诺拒绝；
e) 将数据作为数据参数的数据克隆结果传递至验签方法；
f) 设置 promise 为新的 Promise；
g) 返回 promise 并异步完成余下的步骤；
h) 若余下的步骤或者调用的过程抛出异常或者拒绝承诺，则结束算法；
i) 若 normalizedAlgorithm 的 name 成员不同于密钥的[[algorithm]]内部组织的 name 属性，抛出 InvalidAccessError；
j) 若密钥的[[usages]]内部组织不包含“verify”条目，抛出 InvalidAccessError；
k) 将 result 设置为验签操作的结果，该操作过程指定了特定密钥、算法、签名和消息数据下的 normalizedAlgorithm 验签操作；
l) 将 result 作为 Promise 对象返回。

6.6 杂凑方法

digest 散列方法返回一个新的承诺对象，该对象使用 AlgorithmIdentifier 对数据进行散列操作。它应遵循下列步骤：

a) 将算法参数传递至散列方法；
b) 将数据作为数据参数的 data 克隆数据结果传递至散列方法；
c) 将 normalizedAlgorithm 作为正规化算法的结果，将 alg 设置为 algorithm，将 op 设置为“digest”；

d) 若出错，返回带有 normalizedAlgorithm 的承诺拒绝；
e) 设置 promise 为新的 Promise；
f) 返回 promise 并异步完成余下的步骤；
g) 若余下的步骤或者调用的过程抛出异常或者拒绝承诺，则结束算法；
h) 将 result 设置为散列操作的结果，该操作过程指定了算法和消息数据下的 normalizedAlgorithm 验签操作；
i) 将 result 作为 Promise 对象返回。

6.7 生成密钥方法

调用 generateKey 方法时，应执行如下步骤：

a) 将算法、可提取性和用法分别作为 algorithm、extractable 和 usage 参数传递至生成密钥方法；
b) 将 normalizedAlgorithm 作为正规化算法的结果，将 alg 设置为 algorithm，将 op 设置为“generateKey”；
c) 若出错，返回带有 normalizedAlgorithm 的承诺拒绝；
d) 设置 promise 为新的 Promise；
e) 返回 promise 并异步完成余下的步骤；
f) 若余下的步骤或者调用的过程抛出异常或者拒绝承诺，则结束算法；
g) 将 result 设置为生成密钥操作的结果，该操作过程指定了算法和可提取性和用法下的 normalizedAlgorithm 验签操作；
h) 若结果为 CryptoKey 对象；
i) 且结果的[[type]]内部组成为“secret”或“private”、同时用法 usages 为空，则抛出 SyntaxError；
j) 若结果为 CryptoKeyPair 对象；
k) 且结果的私钥属性的[[usages]]内部组成为空序列，则抛出 SyntaxError；
l) 将 result 作为 Promise 对象返回。

6.8 派生密钥方法

调用 deriveKey 派生密钥方法时，应完成如下步骤：

a) 将 algorithm，baseKey，derivedKeyType，extractable 和 usages 分别作为 algorithm，baseKey，derivedKeyType，extractable 和 keyUsages 参数传递至 deriveKey 方法；
b) 将 normalizedAlgorithm 置为正规化算法的结果，同时 alg 置为 algorithm，op 置为“deriveBits”；
c) 若出错，返回带有 normalizedAlgorithm 的承诺拒绝；
d) 将 normalizedDerivedKeyAlgorithm 作为正规化算法的结果，将 alg 设置为 derivedKeyType，将 op 设置为“importKey”；
e) 若出错，返回带有 normalizedDerivedKeyAlgorithm 的承诺拒绝；
f) 设置 promise 为新的 Promise；
g) 返回 promise 并异步完成余下的步骤；
h) 若余下的步骤或者调用的过程抛出异常或者若余下的步骤或者调用的过程抛出异常或者拒绝承诺返回错误码，则结束算法承诺，则结束算法；
i) 若 normalizedAlgorithm 的 name 成员不同于支持派生操作的注册算法，抛出 NotSupportedError；
j) 若 normalizedDerivedKeyAlgorithm 的 name 成员不同于支持取得密钥长度操作的一个注册算法，抛出 aNotSupportedError；

k) 若 normalizedAlgorithm 的 name 成员不同于 baseKey 的[[algorithm]]内部组成的 name 属性，抛出 InvalidAccessError；
l) 若 baseKey 的[[usages]]内部组成不包含“deriveKey”条目，抛出 InvalidAccessError；
m) 将 length 设置为调用取得密钥长度操作的结果，该操作使用 derivedKeyType 作为参数调用 normalizedDerivedKeyAlgorithm；
n) 将 secret 设置为执行派生比特操作的结果，该操作在 key，algorithm 和 length 参数下调用 normalizedAlgorithm；
o) 将 result 置为执行导入密钥操作的结果。该操作使用“raw”为 format，secret 为 keyData，derivedKeyType 调用 normalizedDerivedKeyAlgorithm；
p) 若 result 的[[type]]内部组成是“secret”或者“private”，同时 usages 为空，则抛出 SyntaxError；
q) 将 result 作为 Promise 对象返回。

6.9 派生比特方法

调用 deriveBits 派生比特方法时，应完成如下步骤：
a) 将 algorithm，baseKey 和 length 分别设置为传递至 deriveBits 方法的 algorithm，baseKey 和 length 参数；
b) 将 normalizedAlgorithm 设置为正规化算法的结果，调用时 alg 设置为 algorithm，op 设置为“deriveBits”；
c) 若出错，返回带有 normalizedAlgorithm 的承诺拒绝；
d) 将 promise 设置为新的 Promise 对象；
e) 返回 promise 并异步完成余下的步骤；
f) 若余下的步骤或者调用的过程抛出异常或者若余下的步骤或者调用的过程抛出异常或者拒绝承诺返回错误码，则结束算法承诺，则结束算法；
g) 若 normalizedAlgorithm 的 name 成员不同于 baseKey 的[[algorithm]]的 name 属性，抛出 InvalidAccessError；
h) 若 baseKey 的[[usages]]内部组成不包含“deriveBits”条目，抛出 InvalidAccessError；
i) 将 result 置为新的 ArrayBuffer，其中包含了执行派生比特操作的结果，其中的 normalizedAlgorithm 使用了 baseKey，algorithm 和 length 参数；
j) 将 result 作为 Promise 对象返回。

6.10 导入密钥方法

调用 importKey 导入密钥方法时，应执行下列步骤：
a) 将 format，algorithm，extractable 和 usages 分别设置为传递至 importKey 方法的 format，algorithm，extractable 和 keyUsages 参数；
b) 将 normalizedAlgorithm 置为正规化算法的结果，其中 alg 设置为 algorithm 且 op 设置为“importKey”；
c) 若出错，返回带有 normalizedAlgorithm 的承诺拒绝；
d) 将 promise 设置为新的 Promise 对象；
e) 返回 promise 并异步完成余下的步骤；
f) 若 format 等于字符串“raw”，“pkcs8”，或者“spki”：
   1) 若传递至 importKey 方法的 keyData 参数是一个 JsonWebKey 字典，抛出 TypeError；
   2) 将 keyData 置为传递至 importKey 方法的 keyData 参数克隆数据的结果。
   若 format 等于字符串“jwk”：

1） 若传递至 importKey 方法的 keyData 参数不是一个 JsonWebKey 字典，抛出 TypeError；
2） 将 keyData 置为传递至 importKey 方法的 keyData 参数。

g） 若余下的步骤或者调用的过程抛出异常或者若余下的步骤或者调用的过程抛出异常或者拒绝承诺返回错误码，则结束算法承诺，则结束算法；
h） 将 result 置为 CryptoKey 对象，该对象是使用 keyData，algorithm，format，extractable 和 usages 指定 normalizedAlgorithm 导入密钥操作的结果；
i） 若 result 的[[type]]内部组成是“secret”或“private”同时 usages 为空，则抛出 SyntaxError；
j） 设置 result 的[[extractable]]内部组成为 extractable；
k） 设置 result 的[[usages]]内部组成为 usage 的正规值；
l） 将 result 作为 Promise 对象返回。

6.11 导出密钥方法

调用 exportKey 导出密钥方法时，应执行下列步骤：

a） 将 format 和 key 分别设置为传递至 exportKey 方法的 format 和 key 参数；
b） 将 promise 设置为新的 Promise 对象；
c） 返回 promise 并异步完成余下的步骤；
d） 若余下的步骤或者调用的过程抛出异常或者若余下的步骤或者调用的过程抛出异常或者拒绝承诺返回错误码，则结束算法；
e） 若 key 的[[algorithm]]内部组成的 name 成员不同于支持导出密钥操作的注册算法，则抛出 NotSupportedError；
f） 若 key 的[[extractable]]内部组成为否，则抛出 InvalidAccessError；
g） 设置 result 为导出密钥操作的结果。该操作由 key 的[[algorithm]]内部组成通过使用密钥值和格式来指定；
h） 将 result 作为 Promise 对象返回。

6.12 封装密钥方法

调用 wraptKey 封装密钥方法时，应执行下列步骤：

a） 将 format，key，wrappingKey 和 algorithm 分别设置为传递至 wrapKey 方法的 format，key，wrappingKey 和 wrapAlgorithm 的参数；
b） 将 normalizedAlgorithm 设置为正规化算法的结果，将 alg 设置为 algorithm 同时将 op 设置为“wrapKey”；
c） 若出错，将 normalizedAlgorithm 设置为正规化算法的结果，将 alg 设置为 algorithm 同时将 op 设置为“encrypt”；
d） 若出错，返回带有 normalizedAlgorithm 的承诺拒绝；
e） 将 promise 设置为新的 Promise 对象；
f） 返回 promise 并异步完成余下的步骤；
g） 若余下的步骤或者调用的过程抛出异常或者若余下的步骤或者调用的过程抛出异常或者拒绝承诺返回错误码，则结束算法；
h） 若 normalizedAlgorithm 的 name 成员不同于支持加密和封装操作的已注册算法，抛出 NotSupportedError；
i） 若 normalizedAlgorithm 的 name 成员不同于 wrappingKey 的[[algorithm]]内部组成的 name 属性，抛出 InvalidAccessError；
j） 若 wrappingKey 的[[usages]]内部组成不包含“wrapKey”条目，抛出 InvalidAccessError；
k） 若 key 的[[algorithm]]内部组成所确定的 algorithm 不支持导出密钥的操作，抛出 NotSup-

portedError；

l) 若 key 的[[extractable]]的内部组成为否，抛出 InvalidAccessError；

m) 将 key 设置为导出密钥操作的的结果，该操作由 key 和 format 指定的[[algorithm]]内部组成来确定；

若 format 不同于字符串“raw”“pkcs8”或者“spki”：

设置 bytes 为 key 值。

若 format 等于字符串“jwk”：

1) 将 key 转换为 ECMAScript 对象。

2) 将 json 设置为符合 JSON 语法的表达密钥的 UTF-16 字符串。

3) 将 bytes 设置为转换 json 得到的字节序列，为一个 JavaScript 字符串。

n) 若 normalizedAlgorithm 支持封装密钥操作：

设置 result 为封装密钥操作的结果，该操作由 algorithm，wrappingKey 密钥和 bytes 明文等数据项所指定的 normalizedAlgorithm 的确定。

否则，

若 normalizedAlgorithm 支持加密操作：

将 result 设置为加密操作的结果，该操作由 algorithm，wrappingKey 密钥和 bytes 明文等数据项指定的 normalizedAlgorithm 来确定。

否则：

抛出 NotSupportedError。

o) 将 result 作为 Promise 对象返回。

6.13 解封密钥方法

调用 unwraptKey 封装密钥方法时，应执行下列步骤：

a) 将 format，unwrappingKey，algorithm，unwrappedKeyAlgorithm，extractable 和 usages 分别设置为传递至解封密钥方法的 format，unwrappingKey，unwrapAlgorithm，unwrappedKeyAlgorithm，extractable 和 keyUsages 的参数；

b) 将 wrappedKey 设置为传递至 unwrap 方法数据参数的数据克隆结果；

c) 将 normalizedAlgorithm 设置为正规化算法的结果，其中 alg 设置为 algorithm 同时 op 设置为“unwrapKey”；

d) 若出错，将 normalizedAlgorithm 设置为正规化算法的结果，其中 alg 设置为 algorithm 同时 op 设置为“decrypt”；

e) 若出错，返回带有 normalizedAlgorithm 的承诺拒绝；

f) 将 normalizedKeyAlgorithm 设置为正规化算法的结果，其中 alg 设置为 unwrappedKeyAlgorithm 同时将 op 设置为“importKey”；

g) 若出错，返回带有 normalizedKeyAlgorithm 的承诺拒绝；

h) 将 promise 设置为新的 Promise 对象；

i) 返回 promise 并异步完成余下的步骤；

j) 若余下的步骤或者调用的过程抛出异常或者若余下的步骤或者调用的过程抛出异常或者拒绝承诺返回错误码，则结束算法；

k) 若 normalizedAlgorithm 的 name 成员不同于 unwrappingKey 的[[algorithm]]内部组成的 name 属性，则抛出 InvalidAccessError；

l) 若[[algorithm]]的[[usages]]内部组成不包含“unwrapKey”条目，则抛出 InvalidAccessError；

m) 若 normalizedAlgorithm 支持解封密钥操作：

将 key 设置为执行解封密钥操作的结果，该过程由 algorithm，unwrappingKey 密钥和 wrappedKey 密文指定的 normalizedAlgorithm 来确定。

否则，

若 normalizedAlgorithm 支持解密操作：

将 key 设置为解密操作的结果，该操作使用 algorithm，unwrappingKey 密钥和 wrappedKey 密文指定的 normalizedAlgorithm 来确定。

否则：

抛出 NotSupportedError。

n) 若 format 为字符串“raw”“pkcs8”或者“spki”：

设置 bytes 为 key 值。

若 format 为字符串“jwk”：

将 bytes 设置为 parseaJWK 算法执行的结果，其中 key 作为 data 进行解析。

o) 将 result 设置为执行导入密钥操作的结果，该操作使用 unwrappedKeyAlgorithm 算法，format，usages 和 extractable，并使用 bytes 作为密钥数据对 normalizedKeyAlgorithm 进行指定；

p) 若 result 的[[type]]内部组成为“secret”或者“private”并且 usages 为空，抛出 SyntaxError；

q) 设置 result 的[[extractable]]内部组成为 extractable；

r) 设置 result 的[[usages]]内部组成为 usages 的正规化值；

s) 将 result 作为 Promise 对象返回。

6.14 异常

SubtleCrypto 接口的方法可以返回错误，抛出预定义的异常而拒绝返回承诺。预定义的异常如表 1 所示。

表 1 预定义的异常

类 型	消 息
NotSupportedError	算法不支持
SyntaxError	缺失所需参数或参数超出范围
InvalidStateError	密钥当前状态下不支持请求的操作
InvalidAccessError	提供的密钥不支持请求的操作
UnknownError	未知原因导致的操作失败(例如内存溢出)
DataError	提供给操作的数据不符合要求
OperationError	操作失败

7 算法流程

7.1 SM3 算法

可识别算法名称为“SM3”，对应接口为杂凑功能 digest。SM3 算法在 GM/T 0004—2012 中定义和规范。

a) 如果 normalizedAlgorithm 的 name 成员与”SM3”字符大小写不敏感匹配的话，令 result 作为输入消息 message 的 SM3 哈希算法(GB/T 32905)的结果。

b) 如果在操作过程中出现了错误，则抛出 OperationError 异常。

c) 返回一个包含 result 的新的 ArrayBuffer。

7.2 SM2 加密算法

7.2.1 描述

可识别算法名称为“SM2”，对应接口包括 generateKey，deriveBits，importKey，exportKey。

7.2.2 生成密钥

生成密钥内容如下。

a) 如果 usages 包含了不是“deriveKey”或“deriveBits”的条目，抛出 SyntaxError 异常。
b) 生成 SM2 椭圆曲线密钥对。
c) 如果在操作过程中出现了错误，抛出 OperationError 异常。
d) 令 algorithm 为新的 KeyAlgorithm 对象。
e) 设置 algorithm 的 name 属性为“SM2-KEYEXCHANGE”。
f) 设置 algorithm 的内部接口[[algorithm]]为 algorithm。
g) 令 publicKey 是与全局对象 this[HTML]相关的新的 CryptoKey，表示了生成密钥对的公钥。
h) 设置 publicKey 的内部接口[[type]]为“public”。
i) 设置 publicKey 的内部接口[[algorithm]]为 algorithm。
j) 设置 publicKey 的内部接口[[extractable]]为 true。
k) 设置 publicKey 的内部接口[[usages]]为空列表。
l) 令 privateKey 是与全局对象 this[HTML]相关的新的 CryptoKey，表示了生成密钥对的私钥。
m) 设置 privateKey 的内部接口[[type]]为“private”。
n) 设置 privateKey 的内部接口[[algorithm]]为 algorithm。
o) 设置 privateKey 的内部接口[[extractable]]为 extractable。
p) 设置 publicKey 的内部接口[[usages]]为 usages 与[“deriveKey”“deriveBits”]的交集。
q) 令 result 为新的 CryptoKeyPair 字典。
r) 设置 result 的 publicKey 属性为 publicKey。
s) 设置 result 的 privateKey 属性为 privateKey。
t) 根据[WEBIDL]中的定义，将 result 转换成 ECMAScript 对象。

7.2.3 派生密钥

派生密钥内容如下。

a) 如果 key 的内部接口[[type]]不是“private”，抛出 InvalidAccessError 异常。
b) 令 publicKey 为 normalizedAlgorithm 的 public 成员。
c) 如果 publicKey 的内部接口[[type]]不是“public”，抛出 InvalidAccessError 异常。
d) 如果 publicKey 的内部接口[[algorithm]]的 name 属性不等于 key 的内部接口[[algorithm]]的 name 属性，抛出 InvalidAccessError 异常。
e) 将 key 为做私钥 d，publicKey 的内部接口[[handle]]作为公钥，执行 SM2 密钥派生操作。
f) 令 secret 为 SM2 密钥派生输出的结果。
g) 如果操作过程中出现了错误，抛出 OperationError 异常。
h) 返回 result。

7.2.4 导入密钥

导入密钥内容如下。

a) 令 keydata 为要导入的密钥数据。

b) 如果 format 为“spki”：

- 如果 usages 不为空，抛出 SyntaxError 异常。
- 令 spki 是对 keyData 运行解析 subjectPublicKeyInfo 算法的结果。
- 如果解析阶段发生了错误，抛出 DataError 异常。
- 如果 spki 的算法标识符域 algorithm 的 algorithm 对象标识符不是“SM2”，抛出 DataError 异常。
- 如果 spki 的算法标识符域 algorithm 的 parameters 域不存在，抛出 DataError 异常。
- 令 publicKey 为使用 spki 的 subjectPublicKey 域执行转换步骤时确定的椭圆曲线的公钥。(非压缩的点格式必须要支持)。
- 如果实现不能支持压缩点格式，同时提供了压缩点，抛出 DataError 异常。
- 如果解析错误发生或者发现了单位点，抛出 DataError 异常。
- 令 key 为与全局对象 this[HTML]相关的新的 CryptoKey，表示 publicKey。
- 如果公钥的值不是 SM2 所规定的椭圆曲线上的合法的点，抛出 DataError 异常。
- 设置 key 的内部接口[[type]]为“public”。
- 令 algorithm 为一个新的 KeyAlgorithm。
- 设置 algorithm 的 name 属性为“SM2-KEYEXCHANGE”。
- 设置 algorithm 为 key 的内部接口[[algorithm]]。

c) 如果 format 是“pkcs8”：

- 如果 usages 包含了不是“deriveKey”或“deriveBits”的条目，抛出 SyntaxError 异常。
- 令 privateKeyInfo 是对 keyData 运行解析 privateKeyInfo 算法的结果。
- 如果解析阶段发生了错误，抛出 DataError 异常。
- 如果 privateKeyInfo 的私钥算法域 privateKeyAlgorithm 的 algorithm 对象标识符不是“SM2”，抛出 DataError 异常。
- 如果 privateKeyInfo 的私钥算法标识符域 privateKeyAlgoritm 的 parameters 域不存在，抛出 DataError 异常。
- 令 sm2PrivateKey 为执行解析 ASN.1 结构体算法的结果，其中 data 为 privateKeyInfo 的 privateKey 域，structure 为 SM2PrivateKey 结构。
- 如果解析过程中发生了错误，抛出 DataError 异常。
- 令 key 为与全局对象 this[HTML]相关的新的 CryptoKey，表示了使用 sm2PrivateKey 的转换步骤中确定的椭圆曲线的私钥。
- 如果私钥的值不是 SM2 所规定的椭圆曲线上的合法的点，抛出 DataError 异常。
- 设置 key 的内部接口[[type]]为“private”。
- 令 algorithm 为一个新的 KeyAlgorithm。
- 设置 algorithm 的 name 属性为“SM2-KEYEXCHANGE”。
- 设置 algorithm 为 key 的内部接口[[algorithm]]。

d) 如果 format 是“jwk”：

- 如果 keyData 是一个 JsonWebKey 字典，令 jwk 是 keyData，否则抛出 DataError 异常。
- 如果 jwk 的“d”域存在，并且 usages 包含了不是“deriveKey”或“deriveBits”的条目，抛出 SyntaxError 异常。
- 如果 jwk 的 d 域不存在，并且 usages 不为空，抛出 SyntaxError 异常。
- 如果 jwk 的“kty”域不是“SM2”，抛出 DataError 异常。
- 如果 usages 不为空，并且 jwk 的“use”域存在，不能字符大小写不敏感匹配于“enc”，抛出 DataError 异常。

- 如果 jwk 的“key_ops”域存在，并且不符合 JsonWebKey 的要求或者不包含规定的所有的 usages 值，抛出 DataError 异常。
- 如果 jwk 的“ext”域存在，并且值为 false，extractable 为 true，抛出 DataError 异常。
- 如果 jwk 的“d”域存在：
  - 如果 jwk 不满足 JSON WEB 算法的 6.4 的需求，抛出 DataError 异常。
  - 令 privateKey 表示根据 JSON web 算法解析 jwk 所识别出的私钥。
  - 如果 privateKey 不是合法的 SM2 私钥，抛出 DataError 异常。
  - 令 key 是一个表示 privateKey 的新的 CrytoKey 对象。
  - 设置 key 的内部接口[[type]]为“private”。
- 否则：
  - 如果 jwk 不满足 JSON WEB 算法的 6.4 的需求，抛出 DataError 异常。
  - 令 publicKey 表示根据 JSON web 算法解析 jwk 所识别出的公钥。
  - 如果 publicKey 不是合法的 SM2 公钥，抛出 DataError 异常。
  - 令 key 是一个表示 publicKey 的新的 CrytoKey 对象。
  - 设置 key 的内部接口[[type]]为“public”。
- 如果密钥值不是 SM2 所规定的椭圆曲线上的合法的点，抛出 DataError 异常。
- 令 algorithm 为一个新的 KeyAlgorithm 字典。
- 设置 algorithm 的 name 属性为“SM2-KEYEXCHANGE”。
- 设置 key 的内部接口[[algorithm]]为 algorithm。

e) 如果 format 是“raw”：
- 如果 usages 不为空，抛出 SyntaxError 异常。
- 令 Q 为在 keydata 上执行 SM2 的转换步骤中确定的椭圆曲线点。（非压缩点格式必须要支持）。
- 如果实现没有支持压缩点格式，但提供了压缩点，抛出 DataError 异常。
- 如果解码错误或者发现了单位点，抛出 DataError 错误。
- 令 key 是与全局对象 this[HTML]相关的新的 CryptoKey，表示了 Q。
- 令 algorithm 为 KeyAlgorithm 对象。
- 设置 algorithm 的 name 属性为“SM2-KEYEXCHANGE”。
- 设置 key 的内部接口[[type]]为“public”。
- 设置 key 的内部接口[[algorithm]]为 algorithm。

f) 如果 foramt 是其他值，抛出 NotSupportedError 异常。

g) 返回 key。

7.2.5 导出密钥

导出密钥内容如下。

a) 令 key 是要导出的 CryptoKey。

b) 如果 key 的内部接口[[handle]]表示的底层密钥资料不能访问，抛出 OperationError 异常。

c) 如果 format 是“spki”：
- 如果 key 的内部接口[[type]]不是“public”，抛出 InvalidAccessError 异常。
- 令 data 是 ASN.1 结构体 subjectPublicKeyInfo 的一个实例。具有下列属性。
  - 设置 algorithm 域为 ASN.1 类型 AlgorithmIdentifier。
  - 设置 algorithm 的 parameter 域为 ASN.1 类型 SM2Parameters，令其 keyData 是表示 key 的内部接口[[handle]]的所代表的椭圆曲线公钥的字符串，使用了非压缩形式。
  - 设置 subjectPublicKey 域为 keyData。

- 令 result 是与全局对象 this[HTML]相关的新的 ArrayBuffer，其中包含了 data。

d) 如果 format 是“pkcs8”：
- 如果 key 的内部接口[[type]]不是“private”，抛出 InvalidAccessError 异常。
- 令 data 是 ASN.1 类型 privateKeyInfo 的实例，具有下列属性：
  - 设置 version 域为 0。
  - 设置 privateKeyAlgorithm 域为 ASN.1 类型 PrivateKeyAlgorithmIdentifier。
  - 设置 parameter 域为 ASN.1 类型 SM2Parameters。令其 keyData 为表示 key 的内部接口[[handle]]的所代表的椭圆曲线公钥的字符串的 SM2PrivateKey 结构体的 DER 编码的实例，并满足下列要求：1）parameters 域存在，并且与该 ASN.1 类型 PrivateKeyInfo 的 privateKeyAlgorithm 域的 parameters 域等价；2）publicKey 域存在，并且与 key 的内部接口[[handle]]代表的椭圆曲线私钥相关的椭圆曲线公钥等价。
  - 设置 privateKey 域为 keyData。
- 令 result 是与全局对象 this[HTML]相关的新的 ArrayBuffer，其中包含了 data。

e) 如果 format 是“jwk”：
- 令 jwk 是一个新的 JsonWebKey 字典
- 设置 jwk 的 kty 属性为字符串“SM2”。
- 根据 JSON WEB 算法的 5.3 中的相应定义设置 jwk 的 x 和 y 属性。
- 如果 key 的内部接口[[type]]为“private”：
  - 根据 JSON WEB 算法的 5.3 中的相应定义设置 jwk 的 d 的属性。
- 设置 jwk 的 key_ops 属性为 key 的 usages 属性。
- 设置 jwk 的 ext 属性为 key 的内部接口[[extractable]]。
- 令 result 为根据[WebIDL]定义的将 jwk 转换成 ECMAScript 对象的结果。

f) 如果 format 是“raw”：
- 如果 key 的内部接口[[type]]不是“public”，抛出 InvalidAccessError 异常。
- 令 data 为 key 的内部接口[[handle]]代表的椭圆曲线点 Q 所表示的字符串，用了非压缩格式。
- 令 result 为与全局对象 this[HTML]相关的新的 ArrayBuffer，其中包含了 data。

g) 如果 format 是其他值，抛出 NotSupportedError 异常。

h) 返回 result。

7.3 SM2 签名算法

7.3.1 描述

可识别算法名称是“SM2-SIGN-VERIFY”。支持的功能包括 sign，verify，generateKey，importKey，exportKey。

7.3.2 签名

签名内容如下。

a) 如果 key 中的内部接口[[type]]不是“private”，抛出 InvalidAccessError 异常。

b) 令 M 是使用 SM3 算法对 message 进行摘要操作的结果。

c) 令 d 是与 key 相关的 SM2 私钥。

d) 令 params 是与 key 相关的 EC 域的参数。

e) 执行 SM2 的签名过程，M 为消息，d 为私钥。

f) 令 r 和 s 为 SM2 签名过程中生成的整数对。

g) 令 result 为与全局对象 this[HTML]相关的新的 ArrayBuffer。

h) 令 n 为使得 n * 8 比椭圆曲线的基点的序的以 2 位底的对数大的最小的整数。
i) 将 r 转换成长度为 n 的字节字符串，并附加到 result 后面。
j) 将 s 转换成长度为 n 的字节字符串，并附加到 result 后面。
k) 返回一个与全局对象 this[HTML]相关的新的 ArrayBuffer，其中包含了 result 的字节数组。

7.3.3 验签

验签内容如下。

a) 如果 key 中的内部接口[[type]]不是“public”，抛出 InvalidAccessError 异常。
b) 令 M 是使用 SM3 算法对 message 进行摘要操作的结果。
c) 令 Q 为与 key 相关的 SM2 公钥。
d) 令 params 是与 key 相关的 SM2 域的参数。
e) 执行 SM2 的验签过程，M 为收到的消息，signature 为收到的签名，Q 为公钥。
f) 当签名为合法时，result 是布尔值 true，否则是 false。
g) 返回 result。

7.3.4 生成密钥

生成密钥内容如下。

a) 如果 usages 包含了不是“sign”或“verify”的条目时，抛出 SyntaxError 异常。
b) 根据 GB/T 32918 的定义，生成 SM2 的一个密钥对。
c) 如果生成密钥对失败，则抛出 OperationError 异常。
d) 令 algorithm 为一个新的 KeyAlgorithm 字典。
e) 将 algorithm 的 name 属性设置为“SM2”。
f) 令 publicKey 是一个与全局对象 this[HTML]相关的新的 CrytoKey 对象，表示了生成密钥对的公钥。
g) 将 publicKey 的内部接口[[type]]设置为“public”。
h) 将 publicKey 的内部接口[[algorithm]]设置为 algorithm。
i) 将 publicKey 的内部接口[[extractable]]设置为 true。
j) 将 publicKey 的内部接口[[usages]]设置为 usages 和“verify”的交集。
k) 令 privateKey 是一个与全局对象 this[HTML]相关的新的 CrytoKey 对象，表示了生成密钥对的私钥。
l) 将 privateKey 的内部接口[[type]]设置为“private”。
m) 将 privateKey 的内部接口[[algorithm]]设置为 algorithm。
n) 将 privateKey 的内部接口[[extractable]]设置为 extractable。
o) 将 privateKey 的内部接口[[usages]]设置为 usages 和“sign”的交集。
p) 令 result 为一个新的 CryptoKeyPair 字典。
q) 将 result 的 publicKey 属性设置为 publicKey。
r) 将 result 的 privateKey 属性设置为 privateKey。
s) 将 result 转换成[WEBIDL]的定义的 ECMAScript 对象，返回 result。

7.3.5 导入密钥

导入密钥内容如下。

a) 令 keyData 是要导入的密钥数据。
b) 如果 format 是“spki”：
   - 如果 usages 包含了不是“verify”的条目，抛出 SyntaxError 异常。

- 令 spki 是运行解析 subjectPublicKeyInfo 算法的结果。
- 如果解析阶段发生了错误，抛出 DataError 异常。
- 如果 spki 的算法标识符域 algorithm 的 algorithm 对象标识符不是“SM2”，抛出 DataError 异常。
- 令 publicKey 为使用 spki 的 subjectPublicKey 域执行转换步骤时确定的椭圆曲线的公钥。(非压缩的点格式必须要支持)。
- 如果实现不能支持压缩点格式，同时提供了压缩点，抛出 DataError 异常。
- 如果解析错误发生或者发现了单位点，抛出 DataError 异常。
- 令 key 为与全局对象 this[HTML]相关的新的 CryptoKey，表示 publicKey。
- 如果公钥的值不是 SM2 所规定的椭圆曲线上的合法的点，抛出 DataError 异常。
- 设置 key 的内部接口[[type]]为“public”。
- 令 algorithm 为一个新的 KeyAlgorithm。
- 设置 algorithm 的 name 属性为“SM2-SIGN-VERIFY”。
- 设置 algorithm 为 key 的内部接口[[algorithm]]。

c) 如果 format 是“pkcs8”：

- 如果 usages 包含了不是“sign”的条目，抛出 SyntaxError 异常。
- 令 privateKeyInfo 是对 keyData 运行解析 privateKeyInfo 算法的结果。
- 如果解析阶段发生了错误，抛出 DataError 异常。
- 如果 privateKeyInfo 的私钥算法域 privateKeyAlgorithm 的 algorithm 对象标识符不是“SM2”，抛出 DataError 异常。
- 令说 sm2PrivateKey 为执行解析 ASN.1 结构体算法的结果，其中 data 为 privateKeyInfo 的 privateKey 域，structure 为 SM2PrivateKey 结构，exactdata 为 true。
- 如果解析过程中发生了错误，抛出 DataError 异常。
- 令 key 为与全局对象 this[HTML]相关的新的 CryptoKey，表示了使用 sm2PrivateKey 的转换步骤中确定的椭圆曲线的私钥。
- 如果私钥的值不是 SM2 所规定的椭圆曲线上的合法的点，抛出 DataError 异常。
- 设置 key 的内部接口[[type]]为“private”。
- 令 algorithm 为一个新的 KeyAlgorithm。
- 设置 algorithm 的 name 属性为“SM2-SIGN-VERIFY”。
- 设置 algorithm 为 key 的内部接口[[algorithm]]。

d) 如果 format 是“jwk”：

- 如果 keyData 是一个 JsonWebKey 字典，令 jwk 是 keyData，否则抛出 DataError 异常。
- 如果 jwk 的“d”域存在，并且 usages 包含了不是“sign”的条目，或者，如果 jwk 的 d 域不存在，并且 usages 包含了额不是“verify”的条目，抛出 SyntaxError 异常。
- 如果 jwk 的“kty”域不能字符大小写不敏感匹配于“SM2”，抛出 DataError 异常。
- 如果 usages 不为空，并且 jwk 的“use”域存在，不能字符大小写不敏感匹配于“sig”，抛出 DataError 异常。
- 如果 jwk 的“key_ops”域存在，并且不符合 JsonWebKey 的要求或者不包含规定的所有的 usages 值，抛出 DataError 异常。
- 如果 jwk 的“ext”域存在，并且值为 false，extractable 为 true，抛出 DataError 异常。
- 如果 jwk 的“d”域存在：
  - 如果 jwk 不满足 JSON WEB 算法的 6.4 节的需求，抛出 DataError 异常。
  - 令 privateKey 表示根据 JSON web 算法解析 jwk 所识别出的私钥。
  - 如果 privateKey 不是合法的 SM2 私钥，抛出 DataError 异常。

- 令 key 是一个表示 privateKey 的新的 CrytoKey 对象。
- 设置 key 的内部接口[[type]]为“private”。

- 否则：
  - 如果 jwk 不满足 JSON WEB 算法的 6.4 节的需求，抛出 DataError 异常。
  - 令 publicKey 表示根据 JSON web 算法解析 jwk 所识别出的公钥。
  - 如果 publicKey 不是合法的 SM2 公钥，抛出 DataError 异常。
  - 令 key 是一个表示 publicKey 的新的 CrytoKey 对象。
  - 设置 key 的内部接口[[type]]为“public”。
- 如果密钥值不是 SM2 所规定的椭圆曲线上的合法的点，抛出 DataError 异常。
- 令 algorithm 为一个新的 KeyAlgorithm 字典。
- 设置 algorithm 的 name 属性为“SM2-SIGN-VERIFY”。
- 设置 key 的内部接口[[algorithm]]为 algorithm。

e) 如果 format 是“raw”：
- 如果 usages 包含了不是“verify”的条目，抛出 SyntaxError 异常。
- 令 Q 为在 keydata 上执行 SM2 的转换步骤中确定的椭圆曲线点。（非压缩点格式必须要支持）。
- 如果实现没有支持压缩点格式，但提供了压缩点，抛出 DataError 异常。
- 解码错误或者发现了单位点，抛出 DataError 错误。
- 令 key 是与全局对象 this[HTML]相关的新的 CryptoKey，表示了 Q。
- 令 algorithm 为 KeyAlgorithm 对象。
- 设置 algorithm 的 name 属性为“SM2”。
- 设置 key 的内部接口[[type]]为“public”。
- 设置 key 的内部接口[[algorithm]]为 algorithm。

f) 如果 foramt 是其他值，抛出 NotSupportedError 异常。

g) 返回 key。

7.3.6 导出密钥

导出密钥内容如下。

a) 令 key 是要导出的 CryptoKey。

b) 如果 key 的内部接口[[handle]]表示的底层密钥资料不能访问，抛出 OperationError 异常。

c) 如果 format 是“spki”：
- 如果 key 的内部接口[[type]]不是“public”，抛出 InvalidAccessError 异常。
- 令 data 是 ASN.1 结构体 subjectPublicKeyInfo 的一个实例。
  - 设置 data 的 algorithm 域为 ASN.1 类型 AlgorithmIdentifier。
  - 设置 algorithm 的 parameter 域为 ASN.1 类型 SM2Parameters，令其 keyData 是表示 key 的内部接口[[handle]]的所代表的椭圆曲线公钥的八字节字符串，使用了非压缩形式。
- 令 result 是与全局对象 this[HTML]相关的新的 ArrayBuffer，其中包含了 data。

d) 如果 format 是“pkcs8”：
- 如果 key 的内部接口[[type]]不是“private”，抛出 InvalidAccessError 异常。
- 令 data 是 ASN.1 类型 privateKeyInfo 的实例，具有下列属性：
  - 设置 version 域为 0。
  - 设置 privateKeyAlgorithm 域为 ASN.1 类型 PrivateKeyAlgorithmIdentifier。
  - 设置 parameter 域为 ASN.1 类型 SM2Parameters。令其 keyData 为表示 key 的内部接口[[handle]]的所代表的椭圆曲线公钥的八字节字符串的 SM2PrivateKey 结构体

的 DER 编码的实例，并满足下列要求：1）parameters 域存在，并且与该 ASN.1 类型 PrivateKeyInfo 的 privateKeyAlgorithm 域的 parameters 域等价；2）publicKey 域存在，并且与 key 的内部接口[[handle]]代表的椭圆曲线私钥相关的椭圆曲线公钥等价。

- 设置 privateKey 域为 keyData。

- 令 result 是与全局对象 this[HTML]相关的新的 ArrayBuffer，其中包含了 data。

e) 如果 format 是“jwk”：

- 令 jwk 是一个新的 JsonWebKey 字典。
- 设置 jwk 的 kty 属性为字符串“SM2”。
- 根据 JSON WEB 算法的 5.3 中的相应定义设置 jwk 的 x 和 y 属性。
- 如果 key 的内部接口[[type]]为“private”：
  - 根据 JSON WEB 算法的 5.3 中的相应定义设置 jwk 的 d 的属性。
- 设置 jwk 的 key_ops 属性为 key 的 usages 属性。
- 设置 jwk 的 ext 属性为 key 的内部接口[[extractable]]。
- 令 result 为根据[WebIDL]定义的将 jwk 转换成 ECMAScript 对象的结果。

f) 如果 format 是“raw”：

- 如果 key 的内部接口[[type]]不是“public”，抛出 InvalidAccessError 异常。
- 令 data 为 key 的内部接口[[handle]]代表的椭圆曲线点 Q 所表示的八字节字符串，用了非压缩格式。
- 令 result 为与全局对象 this[HTML]相关的新的 ArrayBuffer，其中包含了 data。

g) 如果 format 是其他值，抛出 NotSupportedError 异常。

h) 返回 result。

7.4 SM4 算法

7.4.1 描述

“SM4-CBC”算法标识符用来使用 GB/T 32907 规范的 SM4 以 CBC 模式进行加密和解密操作，当在 CBC 模式执行时，不是块大小(128 字节)的整数倍的消息可以进行填充，在密码应用接口中，填充方式仅支持 PCKS#7。

该算法的算法标识名称为“SM4-CBC”。支持的操作为 encrypto，decrytp，generateKey，importKey，exportKey。

7.4.2 加密

加密内容如下。

a) 如果 normalizedAlgorithm 的 iv 成员不是 128 字节长，抛出 OperationError 异常。

b) 令 paddedPlaintext 是添加了填充字节的 ciphertext 内容的结果。

c) 令 ciphertext 是使用 SM4 作为块密码算法执行了 CBC 加密操作后的结果，normalizedAlgorithm 的 iv 成员的内容作为 IVinput 参数，paddedPlaintext 作为输入明文。

d) 返回与全局对象 this[HTML]相关的新的 ArrayBuffer，包含了 ciphertext。

7.4.3 解密

解密内容如下。

a) 如果 normalizedAlgorithm 的 iv 成员不是 128 字节长，抛出 OperationError 异常。

b) 令 paddedPlaintext 是使用 SM4 算法执行了 CBC 解密操作的结果，normalizedAlgorithm 的 iv 成员的内容是 IV 输入参数，ciphertext 的内容为输入密文。

c) 令 p 为 paddedPlaintext 的最后的字节。

d) 如果 p 是 0 或者大于 16，或者如果 paddedPlaintext 的最后 p 个字节任何一个值不是 p，抛出 OperationError 异常。
e) 令 plaintext 是从 paddedPlaintext 的末端删除 p 个字节的结果。
f) 返回与全局对象 this[HTML]相关的新的 ArrayBuffer，包含了 plaintext。

7.4.4 生成密钥

生成密钥内容如下。

a) 如果 usages 包含了不是“encrypt”“decrypt”“wrapKey”或者“unwrapKey”的条目，抛出 SyntaxError。
b) 如果 normalizedAlgorithm 的 length 成员不等于 128，抛出 OperationError。
c) 生成 128 位长的 SM4 密钥。
d) 如果密钥生成步骤失败，抛出 OperationError 异常。
e) 令 key 是新的代表生成的 SM4 密钥的 CryptoKey 对象。
f) 令 algorithm 是新的 SM4KeyAlgorithm。
g) 设置 algorithm 的 name 属性为“SM4-CBC”。
h) 设置 algorithm 的 length 属性为 128。
i) 设置 key 的内部接口[[algorithm]]为 algorithm。
j) 设置 key 的内部接口[[extractable]]为 extractable。
k) 设置 key 的内部接口[[usages]]为 usages。
l) 返回 key。

7.4.5 导入密钥

导入密钥内容如下。

a) 如果 usages 里面包含了不是“encrypt”“decrypt”“wrapKey”或“unwrapKey”的条目，抛出 SyntaxError 异常。
b) 如果 format 是“raw”：
   - 令 data 是包含 keyData 的字符串。
   - 如果 data 的比特长度不是 128，抛出 DataError 异常。
c) 如果 foramt 是“jwk”：
   - 如果 keyData 是 JsonWebKey 字典，令 jwk 为 keyData。
   - 否则抛出 DataError 异常。
   - 如果 jwk 的“kty”域不是“oct”，抛出 DataError 异常。
   - 如果 jwk 不满足 JSON Web 算法的 6.4 的需求，抛出 DataError 异常。
   - 令 data 为 jwk 的“k”域的解码后的结果的字符串。
   - 如果 jwk 的“alg”域存在，并且不是“SM4-CBC”，抛出 DataError 异常。
   - 如果 usages 不为空，并且 jwk 的“use”域存在且不是“enc”，抛出 DataError 异常。
   - 如果 jwk 的“key_ops”域存在，并且根据 JSON Webkey 的需求不合法，或者不包含所有特定的 usages 值，抛出 DataError 异常。
   - 如果 jwk 的“ext”域存在，值为 false，并且 extractable 为 true，抛出 DataError 异常。
d) 如果 foramt 为其他值，抛出 NotSupportedError 异常。
e) 令 key 为代表值为 data 的表示 SM4 密钥的新的 CryptoKey 对象。
f) 令 algorithm 为新的 SM4KeyAlgorithm。
g) 设置 algorithm 的 name 属性为“SM4-CBC”。
h) 设置 algorithm 的 length 属性为 128。

i) 设置 key 的内部接口[[algorithm]]为 algorithm。
j) 返回 key。

7.4.6 导出密钥

导出密钥内容如下。

a) 如果 key 的内部接口[[handle]]所代表的底层密钥资料不可以访问，抛出 OperationError 异常。
b) 如果 format 是“raw”：
   - 令 data 为 key 的内部接口[[handle]]所代表的 key 的原始字节。
   - 令 reuslt 为与全局对象 this[HTML]相关的新的 ArrayBuffer，并包含 data。
c) 如果 foramt 是“jwk”：
   - 如果 keyData 是 JsonWebKey 字典，令 jwk 为 keyData。
   - 否则抛出 DataError 异常。
   - 设置 jwk 的 kty 属性为“oct”。
   - 设置 jwk 的 k 属性为包含 key 的内部接口[[handle]]所代表的 key 的原始字节的字符串，根据 JSON WEB ALGORITHM 的 6.4 进行编码。
   - 设置 jwk 的 alg 属性为“SM4-CBC”。
   - 设置 jwk 的 key_ops 属性为 key 的 usages 属性
   - 设置 jwk 的 ext 属性为 key 的内部接口[[extractable]]。
   - 令 reuslt 为根据[WEBIDL]定义的转换 jwk 为 ECMAScript 对象的结果。
d) 如果 format 为其他值，抛出 NotSupportedError 异常。
e) 返回 result。

7.5 SM4-ECB 算法

7.5.1 描述

“SM4-ECB”算法标识符用来使用 GB/T 32907 规范的 SM4 以 ECB 模式进行加密和解密操作，当在 ECB 模式执行时，不是块大小(128 字节)的整数倍的消息可以进行填充，在密码应用接口中，填充方式仅支持 PCKS#7。

该算法的算法标识名称为“SM4-ECB”。支持的功能包括 encrypt，decrypt，generateKey，importKey，exportKey。

7.5.2 加密

加密内容如下。

a) 令 paddedPlaintext 是添加了填充字节的 ciphertext 内容的结果。
b) 令 ciphertext 是使用 SM4 作为块密码算法执行了 ECB 加密操作后的结果，paddedPlaintext 作为输入明文。
c) 返回与全局对象 this[HTML]相关的新的 ArrayBuffer，包含了 ciphertext。

7.5.3 解密

解密内容如下。

a) 令 paddedPlaintext 是使用 SM4 算法执行了 ECB 解密操作的结果，ciphertext 的内容为输入密文。
b) 令 p 为 paddedPlaintext 的最后的字节。
c) 如果 p 是 0 或者大于 16，或者如果 paddedPlaintext 的最后 p 个字节任何一个值不是 p，抛出 OperationError 异常。

d) 令 plaintext 是从 paddedPlaintext 的末端删除 p 个字节的结果。
e) 返回与全局对象 this[HTML]相关的新的 ArrayBuffer,包含了 plaintext。

7.5.4 生成密钥

生成密钥内容如下。

a) 如果 usages 包含了不是“encrypt”“decrypt”“wrapKey”或者“unwrapKey”的条目,抛出 SyntaxError。
b) 如果 normalizedAlgorithm 的 length 成员不等于 128,抛出 OperationError。
c) 生成 128 位长的 SM4 密钥。
d) 如果密钥生成步骤失败,抛出 OperationError 异常。
e) 令 key 是新的代表生成的 SM4 密钥的 CryptoKey 对象。
f) 令 algorithm 是新的 SM4KeyAlgorithm。
g) 设置 algorithm 的 name 属性为“SM4-ECB”。
h) 设置 algorithm 的 length 属性为 128。
i) 设置 key 的内部接口[[algorithm]]为 algorithm。
j) 设置 key 的内部接口[[extractable]]为 extractable。
k) 设置 key 的内部接口[[usages]]为 usages。
l) 返回 key。

7.5.5 导入密钥

导入密钥内容如下。

a) 如果 usages 里面包含了不是“encrypt”“decrypt”“wrapKey”或“unwrapKey”的条目,抛出 SyntaxError 异常。
b) 如果 format 是“raw”:
   - 令 data 是包含 keyData 的字符串。
   - 如果 data 的比特长度不是 128,抛出 DataError 异常。
c) 如果 foramt 是“jwk”:
   - 如果 keyData 是 JsonWebKey 字典,令 jwk 为 keyData。
   - 否则抛出 DataError 异常。
   - 如果 jwk 的“kty”域不是“oct”,抛出 DataError 异常。
   - 如果 jwk 不满足 JSON Web 算法的 6.4 的需求,抛出 DataError 异常。
   - 令 data 为 jwk 的“k”域的解码后的结果的字符串。
   - 如果 jwk 的“alg”域存在,并且不是“SM4-ECB”,抛出 DataError 异常。
   - 如果 usages 不为空,并且 jwk 的“use”域存在且不是“enc”,抛出 DataError 异常。
   - 如果 jwk 的“key_ops”域存在,并且根据 JSON Webkey 的需求不合法,或者不包含所有特定的 usages 值,抛出 DataError 异常。
   - 如果 jwk 的“ext”域存在,值为 false,并且 extractable 为 true,抛出 DataError 异常。
d) 如果 foramt 为其他值,抛出 NotSupportedError 异常。
e) 令 key 为代表值为 data 的表示 SM4 密钥的新的 CryptoKey 对象。
f) 令 algorithm 为新的 SM4KeyAlgorithm。
g) 设置 algorithm 的 name 属性为“SM4-ECB”。
h) 设置 algorithm 的 length 属性为 128。
i) 设置 key 的内部接口[[algorithm]]为 algorithm。
j) 返回 key。

7.5.6 导出密钥

导出密钥内容如下。

a) 如果 key 的内部接口[[handle]]所代表的底层密钥资料不可以访问，抛出 OperationError 异常。

b) 如果 format 是“raw”：

- 令 data 为 key 的内部接口[[handle]]所代表的 key 的原始字节。
- 令 reuslt 为与全局对象 this[HTML]相关的新的 ArrayBuffer，并包含 data。

c) 如果 foramt 是“jwk”：

- 如果 keyData 是 JsonWebKey 字典，令 jwk 为 keyData。
- 否则抛出 DataError 异常。
- 设置 jwk 的 kty 属性为“oct”
- 设置 jwk 的 k 属性为包含 key 的内部接口[[handle]]所代表的 key 的原始字节的字符串，根据 JSON WEB ALGORITHM 的 6.4 进行编码。
- 设置 jwk 的 alg 属性为“SM4-ECB”。
- 设置 jwk 的 key_ops 属性为 key 的 usages 属性
- 设置 jwk 的 ext 属性为 key 的内部接口[[extractable]]。
- 令 reuslt 为根据[WEBIDL]定义的转换 jwk 为 ECMAScript 对象的结果。

d) 如果 format 为其他值，抛出 NotSupportedError 异常。

e) 返回 result。

中华人民共和国密码
行业标准
浏览器密码应用接口规范
GM/T 0087—2020
*
中国标准出版社出版发行
北京市朝阳区和平里西街甲2号(100029)
北京市西城区三里河北街16号(100045)
网址 www.spc.net.cn
总编室:(010)68533533 发行中心:(010)51780238
读者服务部:(010)68523946
中国标准出版社秦皇岛印刷厂印刷
各地新华书店经销
*
开本 880×1230 1/16 印张 2 字数 52 千字
2021年5月第一版 2021年5月第一次印刷
*
书号:155066·2-35967 定价 28.00 元

GM/T 0087-2020

码上扫一扫 正版服务到