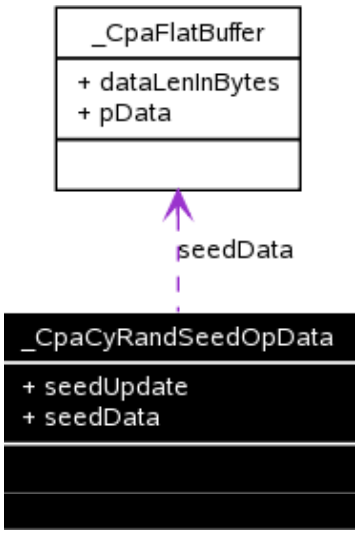
21.6.3 _CpaCyRandSeedOpData Struct Reference

21.6.4 _CpaCyRandSeedOpData 结构引用

Collaboration diagram for _CpaCyRandSeedOpData:

_CpaCyRandSeedOpData 的协作图:



21.6.4.1 Detailed Description

Random Generator Seed Data.

Deprecated:

21.6.4.2 详细描述随机生成器

种子数据。 Deprecated:
As of v1.3 of the API, replaced by **CpaCyDrbgReseedOpData**.
自 API 1.3 版起，由以下内容取代 CpaCyDrbgReseedOpData

This structure lists the different items that required in the cpaCyRandSeed function. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned with the callback.
此结构列出了 cpaCyRandSeed 函数中所需的不同项目。客户端必须为这个结构分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构随回调一起返回时，内存的所有权返回给客户端。

Note:
注意:
If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyRandSeed function, and before it has been returned in the callback, undefined behavior will result.

如果客户端在将此结构中引用的内存提交给 cpaCyRandSeed 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

21.6.4.3 Data Fields
21.6.4.4 数据字段

- **CpaBoolean seedUpdate**
- CpaBoolean seedUpdate
- **CpaFlatBuffer seedData**
- CpaFlatBuffer seedData

21.6.4.5 Field Documentation
21.6.4.6 现场文件

CpaBoolean _CpaCyRandSeedOpData::seedUpdate

When set to CPA_TRUE then the cpaCyRandSeed function will update (combine) the specified seed with the stored seed. When set to CPA_FALSE, the cpaCyRandSeed function will completely discard all existing CpaBoolea_CpaCyRandSeedOpData::seedUpdat entropy in the hardware and replace with the specified seed.
熵并替换为指定的种子。

CpaFlatBuffer _CpaCyRandSeedOpData::seedData

Data for use in either seeding or performing a seed update. The data that is pointed to are random bits and as such do not have an endian order. For optimal performance the data SHOULD be 8-byte aligned. The CpaFlatBuffe_CpaCyRandSeedOpData::seedDat

length of the seed data is in bytes. This MUST currently be equal to CPA_CY_RAND_SEED_LEN_IN_BYTES.

种子数据的长度以字节为单位。这当前必须等于 CPA_CY_RAND_SEED_LEN_IN_BYTES。

21.7 Define Documentation

21.8 定义文档

#define CPA_CY_RAND_SEED_LEN_IN_BYTES

Random Bit/Number Generator Seed Length
随机位/数生成器种子长度

Defines the permitted seed length in bytes that may be used with the cpaCyRandSeed function.
定义可用于 cpaCyRandSeed 函数的允许种子长度(以字节为单位)。

See also:
另请参见:
cpaCyRandSeed
cpaCyRandSeed

21.9 Typedef Documentation

21.10 Typedef 文档

typedef struct _CpaCyRandStats CPA_DEPRECATED

Random Data Generator Statistics.
随机数据生成器统计。

Deprecated:
Deprecated:
As of v1.3 of the API, replaced by **CpaCyDrbgStats64**.
自 API 1.3 版起，由以下内容取代 CpaCyDrbgStats64

This structure contains statistics on the random data generation operations. Statistics are set to zero when the component is initialized, and are collected per instance.
此结构包含随机数据生成操作的统计信息。当组件初始化时，统计信息被设置为零，并针对每个实例进行收集。

typedef struct _CpaCyRandGenOpData CPA_DEPRECATED

Random Bit/Number Generation Data.
随机位/数生成数据。

Deprecated:

Deprecated:

As of v1.3 of the API, replaced by **CpaCyDrbgGenOpData**.

自 API 1.3 版起，由以下内容取代 CpaCyDrbgGenOpData

This structure lists the different items that are required in the cpaCyRandGen function. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned with the callback.

此结构列出了 cpaCyRandGen 函数中所需的不同项目。客户端必须为这个结构分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构随回调一起返回时，内存的所有权返回给客户端。

Note:

注意：

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyRandGen function, and before it has been returned in the callback, undefined behavior will result.

如果客户端在将此结构中引用的内存提交给 cpaCyRandGen 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

typedef struct _CpaCyRandSeedOpData CPA_DEPRECATED

Random Generator Seed Data.

随机生成器种子数据。

Deprecated:

Deprecated:

As of v1.3 of the API, replaced by **CpaCyDrbgReseedOpData**.

自 API 1.3 版起，由以下内容取代 CpaCyDrbgReseedOpData

This structure lists the different items that required in the cpaCyRandSeed function. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned with the callback.

此结构列出了 cpaCyRandSeed 函数中所需的不同项目。客户端必须为这个结构分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构随回调一起返回时，内存的所有权返回给客户端。

Note:

注意：

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyRandSeed function, and before it has been returned in the callback, undefined behavior will result.

如果客户端在将此结构中引用的内存提交给 cpaCyRandSeed 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

21.10 Function Documentation

21.11 功能文档

```
                                                                    instanceHandle,
cpaCyRandGen                        ( const                         pRandGenCb,
                                    const
                                      void *pCallbackTag,
                                      const struct   pRandGenOpData,
                                      *
                                      *                             pRandData
                                    )
```

Random Bits or Number Generation Function.
随机位或数字生成功能。

Deprecated:
Deprecated:

As of v1.3 of the API, replaced by **cpaCyDrbgGen()**.
自 API 1.3 版起，由以下内容取代 cpaCyDrbgGen()

This function is used to request the generation of random bits or a random number. The generated data and the length of the data will be returned to the caller in an asynchronous callback function. If random number generation is selected, the random bits generated by the hardware will be converted to a random number that is compliant to the ANSI X9.82 Part 1 specification.
该函数用于请求生成随机位或随机数。生成的数据和数据长度将在异步回调函数中返回给调用者。如果选择随机数生成，硬件生成的随机位将被转换为符合 ANSI X9.82 第 1 部分规范的随机数。

Context:
背景:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:
None
没有人

Blocking:
阻止:
Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入:
No
不

Thread-safe:
线程安全:
Yes
是

Parameters:
参数:

[in] *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。

[in] *pRandGenCb* Pointer to callback function to be invoked when the operation is complete. If this is set to a NULL value the function will operate synchronously.
[in] pRandGenCb 当作业为时，要叫用的回呼函式指标
完成。如果设置为空值，函数将同步运行。

[in] *pCallbackTag* Opaque User Data for this specific call. Will be returned unchanged in the callback.
[in] pCallbackTag 此特定调用的不透明用户数据。将在回调中不变地返回。

[in] *pRandGenOpData* Structure containing all the data needed to perform the random bit/number operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is
[in] pRandGenOpData 结构，包含执行随机所需的所有资料
位/数运算。客户端代码为此结构分配内存。该组件取得内存的所有权，直到

returned in the callback.

在回调中返回。

[out] *pRandData* Pointer to the memory allocated by the client where the random data will be written to. For optimal performance, the data pointed to SHOULD be 8-byte aligned. There is no endianness associated with the random data. On invocation the callback function will contain this parameter in the pOut parameter.

[out]指向客户端分配的内存的 pRandData 指针，随机数据将写入该内存。为了获得最佳性能，指向的数据应该 8 字节对齐。没有与随机数据相关联的字符顺序。在调用时，回调函数将在 pOut 参数中包含这个参数。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_FAIL 函数失败。

CPA_STATUS_RETRY Resubmit the request.

CPA_STATUS_RETRY 重新提交请求。

CPA_STATUS_INVALID_PARAM Invalid parameter passed in.

传递的 CPA_STATUS_INVALID_PARAM 参数无效。

CPA_STATUS_RESOURCE Error related to system resources. One reason may be for an

与系统资源相关的 CPA_STATUS_RESOURCE 错误。一个原因可能是

entropy test failing.

熵测试失败。

CPA_STATUS_UNSUPPORTED Function is not supported.

不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.

该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

When pRandGenCb is	non-NULL an asynchronous callback of type CpaCyRandGenCbFunc is generated in response to this function call. Any errors generated during processing are reported as part of the callback status code. Entropy testing and reseeding are performed automatically by this function. 当 pRandGenCb 为非 NULL 时，将生成一个 CpaCyRandGenCbFunc 类型的异步回调来响应此函数调用。处理过程中产生的任何错误都会作为回调状态代码的一部分进行报告。熵测试和重新播种是由这个函数自动执行的。

See also:

另请参见:

CpaCyGenFlatBufCbFunc, CpaCyRandGenOpData, **cpaCyRandSeed()**.

CpaCyGenFlatBufCbFunc，CpaCyRandGenOpData，cpaCyRandSeed()

```
cpaCyRandSeed                          ( const  const  void *
                                       const struct
                                       *
                                       )
                                                                instanceHandle, Callback
                                                                pRandSeedCb, pCallbackT
                                                                pSeedOpData
```

Random Data Generator Seed Function.

随机数据发生器种子函数。

Deprecated:

Deprecated:

As of v1.3 of the API, replaced by **cpaCyDrbgReseed()**.

自 API 1.3 版起，由以下内容取代 cpaCyDrbgReseed()

This function is used to either seed or perform a seed update on the random data generator. Replacing the seed with a user supplied seed value, or performing a seed update are completely optional operations. If seeding is specified, it has the effect or disregarding all existing entropy within the random data generator and replacing with the specified seed. If performing a seed update, then the specified seed is mixed into the stored seed. The seed length MUST be equal to CPA_CY_RAND_SEED_LEN_IN_BYTES.

该函数用于在随机数据生成器上播种或执行种子更新。用用户提供的种子值替换种子或执行种子更新完全是可选操作。如果指定了种子设定，则会忽略随机数据生成器中的所有现有熵，并替换为指定的种子。如果执行种子更新，则将指定的种子混合到存储的种子中。种子长度必须等于 CPA_CY_RAND_SEED_LEN_IN_BYTES。

Context:

背景:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:

假设:

None

没有人

Side-Effects:
副作用:
None
没有人

Blocking:
阻止:
Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入:
No
不

Thread-safe:
线程安全:
Yes
是

Parameters:
参数:
[in] *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。
[in] *pRandSeedCb* Pointer to callback function to be invoked when the operation is complete. If this is set to a NULL value the function will operate synchronously.
[in] pRandSeedCb 指标，指向作业完成时要叫用的回呼函式。如果设置为空值，函数将同步运行。
[in] *pCallbackTag* Opaque User Data for this specific call. Will be returned unchanged in the callback.
[in] pCallbackTag 此特定调用的不透明用户数据。将在回调中不变地返回。
[in] *pSeedOpData* Structure containing all the data needed to perform the random generator seed operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
[in]包含执行随机生成器种子操作所需的所有数据的 pSeedOpData 结构。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

Return values:

返回值:
CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。
CPA_STATUS_FAIL Function failed.
CPA_STATUS_FAIL 函数失败。
CPA_STATUS_RETRY Resubmit the request.
CPA_STATUS_INVALID_PARAM Invalid parameter passed in.
CPA_STATUS_RESOURCE Error related to system resources.
CPA_STATUS_UNSUPPORTED Function is not supported.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.
该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:

后置条件:

None
没有人

Note:

注意:

When pRandSeedCn is	non-NULL an asynchronous callback of type CpaCyRandSeedCbFunc is generated in response to this function call. Any errors generated during processing are reported as part of the callback status code. Entropy testing and reseeding are performed automatically by the cpaCyRandGen function. 当 pRandSeedCn 为非 NULL 时，会生成一个 CpaCyRandSeedCbFunc 类型的异步回调来响应此函数调用。处理过程中产生的任何错误都会作为回调状态代码的一部分进行报告。熵测试和重新播种是由 cpaCyRandGen 函数自动执行的。

See also:

另请参见:

CpaCyGenericCbFunc, CpaCyRandSeedOpData, **cpaCyRandGen()**
CpaCyGenericCbFunc, CpaCyRandSeedOpData, cpaCyRandGen()

CpaStatus CPA_DEPRECATED cpaCyRandQueryStats (const **CpaInstanceHandle**
CpaStatus CPA_DEPRECATED cpaCyRandQueryStats(const CpaInstanceHandle *instanceHandle*,
struct **CpaCyRandStats** * *pRandStats*
)

Query random number statistics specific to an instance.

查询特定于实例的随机数统计信息。

Deprecated:

Deprecated:

As of v1.3 of the API, replaced by **cpaCyDrbgQueryStats64()**.

自 API 1.3 版起，由以下内容取代 cpaCyDrbgQueryStats64()

This function will query a specific instance for random number statistics. The user MUST allocate the CpaCyRandStats structure and pass the reference to that into this function call. This function will write the statistic results into the passed in CpaCyRandStats structure.

该函数将查询特定实例的随机数统计信息。用户必须分配 CpaCyRandStats 结构，并将对该结构的引用传递给这个函数调用。该函数将把统计结果写入传入的 CpaCyRandStats 结构中。

Note: statistics returned by this function do not interrupt current data processing and as such can be slightly out of sync with operations that are in progress during the statistics retrieval process.

注意:此函数返回的统计数据不会中断当前的数据处理，因此可能会与统计数据检索过程中正在进行的操作稍微不同步。

Context:

背景:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

这是一个同步功能，它可以休眠。它不能在不允许休眠的上下文中执行。

Assumptions:

假设:

None

没有人

Side-Effects:

副作用:

None

没有人

Blocking:

阻止:

This function is synchronous and blocking.

这个函数是同步的和阻塞的。

Reentrant:

可重入:

No

不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

[in] *instanceHandle* Instance handle.

[in] instanceHandle 执行个体控制代码。

[out] *pRandStats* Pointer to memory into which the statistics will be written.

[out] pRandStats 指向将写入统计信息的内存的指针。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_INVALID_PARAM Invalid parameter passed in.

CPA_STATUS_RESOURCE Error related to system resources.

CPA_STATUS_UNSUPPORTED Function is not supported.

CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

Component has been initialized.

组件已初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

This function operates in a synchronous manner and no asynchronous callback will be generated.

该函数以同步方式运行，不会生成异步回调。

See also:

另请参见:

CpaCyRandStats

CpaCyRandStats

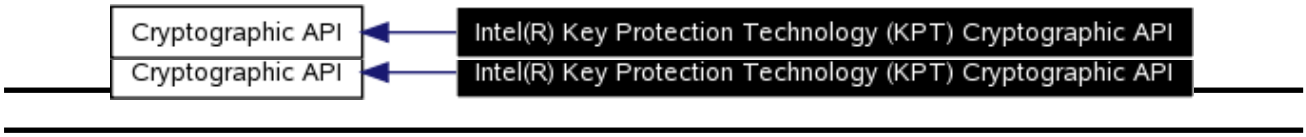
22 Intel(R) Key Protection Technology (KPT) Cryptographic API

23 英特尔密钥保护技术(KPT)加密 API

[Cryptographic API]

[Cryptographic API]

Collaboration diagram for Intel(R) Key Protection Technology (KPT) Cryptographic API:
英特尔密钥保护技术(KPT)加密 API 的协作图:



23.1 Detailed Description

23.2 详细描述

File: cpa_cy_kpt.h

文件:cpa_cy_kpt.h

These functions specify the APIs for Key Protection Technology (KPT) Cryptographic services.
这些函数指定了密钥保护技术(KPT)加密服务的 API。

Note:
注意:

These functions implement the KPT Cryptographic API, In order to realize full KPT function, you need Intel(R) PTT (Platform Trust Technology) and Intel C62X PCH support, which provide 1. QuickAssist Technology 2. Trusted Platform Module (TPM2.0) 3. Secure communication channel between QAT and PTT
这些功能实现了 KPT 加密 API，为了实现完整的 KPT 功能，需要英特尔(R) PTT(平台信任技术)和英特尔 C62X PCH 的支持，它们提供了 1。快速辅助技术 2。可信平台模块(TPM2.0) 3。QAT 和 PTT 之间的安全通信信道

23.3 Data Structures

23.4 数据结构

- struct **CpaCyKptWrappingFormat_t**
- 结构体 CpaCyKptWrappingFormat_t

- struct **CpaCyKptRsaWpkSizeRep2_t**
- 结构体 CpaCyKptRsaWpkSizeRep2_t
- union **CpaCyKptWpkSize_t**
- 联盟 CpaCyKptWpkSize_t
- struct **CpaCyKptUnwrapContext_t**
- 结构体 CpaCyKptUnwrapContext_t
- struct **_CpaCyKptEcdsaSignRSOpData**
- 结构体_CpaCyKptEcdsaSignRSOpData

23.5 Defines
23.6 界定

- #define **CPA_CY_KPT_MAX_IV_LENGTH**

- #定义 CPA_CY_KPT_MAX_IV_LENGTH
- #define **CPA_CY_KPT_HMAC_LENGTH**
- #定义 CPA_CY_KPT_HMAC_LENGTH
- #define **CPA_CY_KPT_CALLER_NONCE_LENGTH**
- #定义 CPA_CY_KPT_CALLER_NONCE_LENGTH
- #define **CPA_CY_KPT_DEVICE_NONCE_LENGTH**
- #定义 CPA_CY_KPT_DEVICE_NONCE_LENGTH

23.7 Typedefs
23.8 类型定义

- typedef **Cpa64U CpaCyKptHandle**

- 数据类型说明 Cpa64U CpaCyKptHandle
- typedef enum **CpaCyKptWrappingKeyType_t CpaCyKptWrappingKeyType**
- typedef 枚举 CpaCyKptWrappingKeyType_t CpaCyKptWrappingKeyType
- typedef enum **CpaCyKptHMACType_t CpaCyKptHMACType**
- typedef 枚举 CpaCyKptHMACType_t CpaCyKptHMACType
- typedef enum **CpaCyKptKeyManagementStatus_t CpaCyKptKeyManagementStatus**
- typedef 枚举 CpaCyKptKeyManagementStatus_t CpaCyKptKeyManagementStatus
- typedef enum **CpaCyKptKeySelectionFlags_t CpaCyKptKeySelectionFlags**
- typedef 枚举 CpaCyKptKeySelectionFlags_t CpaCyKptKeySelectionFlags
- typedef enum **CpaCyKptKeyAction_t CpaCyKptKeyAction**
- typedef 枚举 CpaCyKptKeyAction_t CpaCyKptKeyAction
- typedef **CpaCyKptWrappingFormat_t CpaCyKptWrappingFormat**
- 数据类型说明 CpaCyKptWrappingFormat_t CpaCyKptWrappingFormat
- typedef **CpaCyKptRsaWpkSizeRep2_t CpaCyKptRsaWpkSizeRep2**
- 数据类型说明 CpaCyKptRsaWpkSizeRep2_t CpaCyKptRsaWpkSizeRep2
- typedef **CpaCyKptWpkSize_t CpaCyKptWpkSize**
- 数据类型说明 CpaCyKptWpkSize_t CpaCyKptWpkSize
- typedef **CpaCyKptUnwrapContext_t CpaCyKptUnwrapContext**
- 数据类型说明 CpaCyKptUnwrapContext_t CpaCyKptUnwrapContext
- typedef **_CpaCyKptEcdsaSignRSOpData CpaCyKptEcdsaSignRSOpData**
- 数据类型说明_CpaCyKptEcdsaSignRSOpData CpaCyKptEcdsaSignRSOpData

23.9 Enumerations

23.10 列举

22.5 Enumerations

22.6 列举

- enum **CpaCyKptWrappingKeyType_t** {
  CPA_CY_KPT_WRAPPING_KEY_TYPE_AES128_GCM,
  CPA_CY_KPT_WRAPPING_KEY_TYPE_AES256_GCM,
  CPA_CY_KPT_WRAPPING_KEY_TYPE_AES128_CBC,
  CPA_CY_KPT_WRAPPING_KEY_TYPE_AES256_CBC

- 列举型别 CpaCyKptWrappingKeyType_t
  }
  }
- enum **CpaCyKptHMACType_t** {
  CPA_CY_KPT_HMAC_TYPE_NULL,
  CPA_CY_KPT_HMAC_TYPE_SHA1,
  CPA_CY_KPT_HMAC_TYPE_SHA224,
  CPA_CY_KPT_HMAC_TYPE_SHA256,
  CPA_CY_KPT_HMAC_TYPE_SHA384,
  CPA_CY_KPT_HMAC_TYPE_SHA512,
  CPA_CY_KPT_HMAC_TYPE_SHA3_224,
  CPA_CY_KPT_HMAC_TYPE_SHA3_256,
  CPA_CY_KPT_HMAC_TYPE_SHA3_384,
  CPA_CY_KPT_HMAC_TYPE_SHA3_512
- 列举型别 CpaCyKptHMACType_t
  CPA_CY_KPT_HMAC_TYPE_NULL
  }
  }
- enum **CpaCyKptKeyManagementStatus_t** {
  CPA_CY_KPT_SUCCESS,
  CPA_CY_KPT_REGISTER_HANDLE_FAIL_RETRY,
  CPA_CY_KPT_REGISTER_HANDLE_FAIL_DUPLICATE,
  CPA_CY_KPT_LOAD_KEYS_FAIL_INVALID_HANDLE,
  CPA_CY_KPT_REGISTER_HANDLE_FAIL_WKT_FULL,
  CPA_CY_KPT_WKT_ENTRY_NOT_FOUND,
- 列举型别 CpaCyKptKeyManagementStatus_t
  CPA_CY_KPT_SUCCESSCPA_CY_KPT_REGISTER_HANDLE_FAIL_RE
  TRYCPA_CY_KPT_REGISTER_HANDLE_FAIL_DUPLICATECPA_CY_K
  PT_LOAD_KEYS_FAIL_INVALID_HANDLECPA_CY_KPT_REGISTER_
  HANDLE_FAIL_WKT_FULLCPA_CY_KPT_WKT_ENTRY_NOT_FOUND
  CPA_CY_KPT_REGISTER_HANDLE_FAIL_INSTANCE_QUOTA_EXCEEDED,
  CPA_CY_KPT_LOADKEYS_FAIL_CHECKSUM_ERROR,
  CPA_CY_KPT_LOADKEYS_FAIL_HANDLE_NOT_REGISTERED,
  CPA_CY_KPT_LOADKEYS_FAIL_POSSIBLE_DOS_ATTACK,
  CPA_CY_KPT_LOADKEYS_FAIL_INVALID_AC_SEND_HANDLE,
  CPA_CY_KPT_LOADKEYS_FAIL_INVALID_DATA_OBJ,
  CPA_CY_KPT_FAILED
  CPA_CY_KPT_REGISTER_HANDLE_FAIL_INSTANCE_QUOTA_EXCEEDED,CPA_CY_KPT_LOADK
  EYS_FAIL_CHECKSUM_ERRORCPA_CY_KPT_LOADKEYS_FAIL_HANDLE_NOT_REGISTEREDCPA
  _CY_KPT_LOADKEYS_FAIL_POSSIBLE_DOS_ATTACKCPA_CY_KPT_LOADKEYS_FAIL_INVALI
  D_AC_SEND_HANDLECPA_CY_KPT_LOADKEYS_FAIL_INVALID_DATA_OBJ
  }
  }
- enum **CpaCyKptKeySelectionFlags_t** {
  CPA_CY_KPT_SWK,
  CPA_CY_KPT_WPK,

CPA_CY_KPT_OPAQUE_DATA,
CPA_CY_KPT_HMAC_AUTH_PARAMS,
CPA_CY_KPT_RN_SEED

- 列举型别 CpaCyKptKeySelectionFlags_t
  CPA_CY_KPT_SWKCPA_CY_KPT_WPKCPA_CY_KP
  T_OPAQUE_DATACPA_CY_KPT_HMAC_AUTH_PAR
  AMSCPA_CY_KPT_RN_SEED
  }
  }
- enum **CpaCyKptKeyAction_t** {
  CPA_CY_KPT_NO_HMAC_AUTH_CHECK,
  CPA_CY_KPT_HMAC_AUTH_CHECK
- 列举型别 CpaCyKptKeyAction_t
  CPA_CY_KPT_NO_HMAC_AUTH_CHECKCPA_CY_KPT
  _HMAC_AUTH_CHECK
  }
  }

22.7 Functions
22.8 功能

- **CpaStatus cpaCyKptRegisterKeyHandle** (**CpaInstanceHandle** instanceHandle, **CpaCyKptHandle**

- CpaStatus cpaCyKptRegisterKeyHandle （CpaInstanceHandle CpaCyKptHandle keyHandle, **CpaCyKptKeyManagementStatus** *pKptStatus)
  钥匙把手，CpaCyKptKeyManagementStatus
- **CpaStatus cpaCyKptLoadKeys** (**CpaInstanceHandle** instanceHandle, **CpaCyKptHandle** keyHandle, **CpaCyKptWrappingFormat** *pKptWrappingFormat, **CpaCyKptKeySelectionFlags** keySelFlag, **CpaCyKptKeyAction** keyAction, **CpaFlatBuffer** *pOutputData, **CpaCyKptKeyManagementStatus** *pKptStatus)
- CpaStatus cpaCyKptLoadKeys （CpaInstanceHandle CpaCyKptHandleCpaCyKptWrappingFormat CpaCyKptKeySelectionFlagsCpaCyKptKeyAction CpaFlatBuffer CpaCyKptKeyManagementStatus

22.6 Functions

22.7 功能

- **CpaStatus cpaCyKptDeleteKey** (**CpaInstanceHandle** instanceHandle, **CpaCyKptHandle**
- CpaStatus cpaCyKptDeleteKey （CpaInstanceHandle CpaCyKptHandle keyHandle, **CpaCyKptKeyManagementStatus** *pkptstatus)
  钥匙把手，CpaCyKptKeyManagementStatus
- **CpaStatus cpaCyKptRsaDecrypt** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyKptRsaDecrypt （常量 CpaInstanceHandle **CpaCyGenFlatBufCbFunc** pRsaDecryptCb, void *pCallbackTag, const **CpaCyRsaDecryptOpData** CpaCyGenFlatBufCbFunc pRsaDecryptCb，void *pCallbackTag，constCpaCyRsaDecryptOpData *pDecryptOpData, **CpaFlatBuffer** *pOutputData, **CpaFlatBuffer** *pKptUnwrapContext)
  *pDecryptOpData，CpaFlatBuffer CpaFlatBuffer
- **CpaStatus cpaCyKptEcdsaSignRS** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyKptEcdsaSignRS （常量 CpaInstanceHandle **CpaCyEcdsaSignRSCbFunc** pCb, void *pCallbackTag, const **CpaCyKptEcdsaSignRSOpData** CpaCyEcdsaSignRSCbFunc pCb，void *pCallbackTag，constCpaCyKptEcdsaSignRSOpData *pOpData, **CpaBoolean** *pSignStatus, **CpaFlatBuffer** *pR, **CpaFlatBuffer** *pS, **CpaFlatBuffer** *pOpData，CpaBoolean CpaFlatBuffer CpaFlatBuffer CpaFlatBuffer *pKptUnwrapContext)
  *pKptUnwrapContext)
- **CpaStatus cpaCyKptDsaSignS** (const **CpaInstanceHandle** instanceHandle, const **CpaCyDsaGenCbFunc** pCb, void *pCallbackTag, const **CpaCyDsaSSignOpData** *pOpData, **CpaBoolean** *pProtocolStatus, **CpaFlatBuffer** *pS, **CpaFlatBuffer** *pKptUnwrapContext)
- CpaStatus cpaCyKptDsaSignS （常量 CpaInstanceHandle CpaCyDsaGenCbFunc CpaCyDsaSSignOpData CpaBoolean CpaFlatBuffer CpaFlatBuffer
- **CpaStatus cpaCyKptDsaSignRS** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyKptDsaSignRS （常量 CpaInstanceHandle **CpaCyDsaRSSignCbFunc** pCb, void *pCallbackTag, const **CpaCyDsaRSSignOpData** *pOpData, CpaCyDsaRSSignCbFunc pCb，void *pCallbackTag，constCpaCyDsaRSSignOpData **CpaBoolean** *pProtocolStatus, **CpaFlatBuffer** *pR, **CpaFlatBuffer** *pS, **CpaFlatBuffer** CpaBoolean *协议状态，CpaFlatBuffer CpaFlatBuffer CpaFlatBuffer *pKptUnwrapContext)
  *pKptUnwrapContext)

22.8 Data Structure Documentation

22.9 数据结构文档

22.9.1 CpaCyKptWrappingFormat_t Struct Reference

22.9.2 CpaCyKptWrappingFormat_t 结构引用

22.9.2.1 Detailed Description
22.9.2.2 详细描述

KPT wrapping format structure.
KPT 包装格式结构。

This structure defines wrapping format which is used to wrap clear private keys using a symmetric wrapping key.Application sets these parameters through the cpaCyKptLoadKeys calls.
该结构定义了包装格式，用于使用对称包装密钥包装明文私钥。应用程序通过调用 cpaCyKptLoadKeys 来设置

这些参数。

22.9.2.3 Data Fields
22.9.2.4 数据字段

- **CpaCyKptWrappingKeyType wrappingAlgorithm**
- CpaCyKptWrappingKeyType wrappingAlgorithm
- **Cpa8U iv** [CPA_CY_KPT_MAX_IV_LENGTH]
- Cpa8U iv [CPA_CY_KPT_MAX_IV_LENGTH]
- **Cpa32U iterationCount**
- Cpa32U iterationCount
- **CpaCyKptHMACType hmacType**
- CpaCyKptHMACType hmacType

22.9.2.5 Field Documentation
22.9.2.6 现场文件

Symmetric wrapping algorithm
对称包装算法

Initialization Vector
初始化向量

Iteration Count for Key Wrap Algorithms
密钥包装算法迭代计数

Hash algorithm used in WPK tag generation
WPK 标签生成中使用的哈希算法

22.7.1 CpaCyKptWrappingFormat_t Struct Reference

22.7.2CpaCyKptWrappingFormat_t 结构引用

22.7.3 CpaCyKptRsaWpkSizeRep2_t Struct Reference

22.7.4 CpaCyKptRsaWpkSizeRep2_t 结构引用

22.7.4.1 Detailed Description
22.7.4.2 详细描述

RSA wrapped private key size structure For Representation 2.
表示 2 的 RSA 包装私钥大小结构。

This structure contains byte length of wrapped quintuple of p,q,dP,dQ and qInv which are required for the second representation of RSA private key. PKCS #1 V2.1 specification defines the second representation of the RSA private key,The quintuple of p, q, dP, dQ, and qInv are required for this representation.
此结构包含包装的五元组 p、q、dP、dQ 和 qInv 的字节长度，这是 RSA 私钥的第二种表示法所需要的。PKCS #1 V2.1 规范定义了 RSA 私钥的第二种表示，这种表示需要 p、q、dP、dQ 和 qInv 的五元组。
CpaCyRsaPrivateKeyRep2
CpaCyRsaPrivateKeyRep2

22.7.4.3 Data Fields
22.7.4.4 数据字段

- **Cpa32U pLenInBytes**
- Cpa32U pLenInBytes
- **Cpa32U qLenInBytes**
- Cpa32U qLenInBytes
- **Cpa32U dpLenInBytes**
- Cpa32U dpLenInBytes
- **Cpa32U dqLenInBytes**
- Cpa32U dqLenInBytes
- **Cpa32U qinvLenInBytes**
- Cpa32U qinvLenInBytes

22.7.4.5 Field Documentation
22.7.4.6 现场文件

Cpa32U CpaCyKptRsaWpkSizeRep2_t::pLenInBytes

The byte length of wrapped prime p
包装素数 p 的字节长度

Cpa32U CpaCyKptRsaWpkSizeRep2_t::qLenInBytes

The byte length of wrapped prime q
包装质数 q 的字节长度

Cpa32U CpaCyKptRsaWpkSizeRep2_t::dpLenInBytes

The byte length of wrapped factor CRT exponent (dP)
包装因子 CRT 指数(dP)的字节长度

The byte length of wrapped factor CRT exponent (dQ)

包装因子 CRT 指数(dQ)的字节长度

The byte length of wrapped coefficient (qInv)

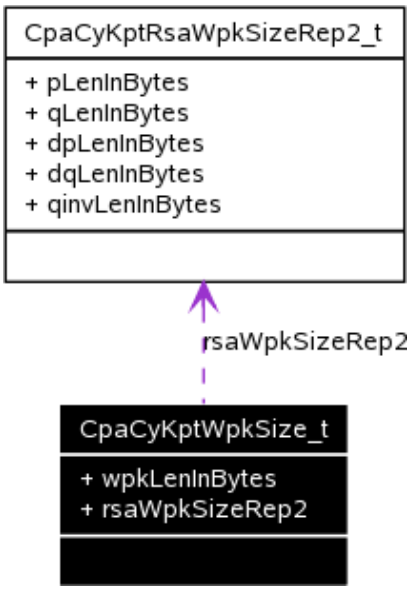
包裹系数的字节长度(qInv)

22.7.5 CpaCyKptWpkSize_t Union Reference

22.7.6 CpaCyKptWpkSize_t 联合引用

Collaboration diagram for CpaCyKptWpkSize_t:

CpaCyKptWpkSize_t 的协作图:



22.7.6.1 Detailed Description

22.7.6.2 详细描述

Wrapped private key size union.
包装私钥大小联合。

A wrapped private key size union, either wrapped quintuple of RSA representation 2 private key,or byte length of wrapped ECC/RSA Rep1/DSA/ ECDSA private key.
包装的私钥大小联合，或者是 RSA 表示 2 私钥的包装五元组，或者是包装的 ECC/RSA Rep1/DSA/ ECDSA 私钥的字节长度。

22.7.6.3 Data Fields
22.7.6.4 数据字段

- **Cpa32U wpkLenInBytes**
- Cpa32U wpkLenInBytes
- **CpaCyKptRsaWpkSizeRep2 rsaWpkSizeRep2**
- CpaCyKptRsaWpkSizeRep2 rsaWpkSizeRep2

22.7.6.5 Field Documentation
22.7.6.6 现场文件

Cpa32U CpaCyKptWpkSize_t::wpkLenInBytes
Cpa32CpaCyKptWpkSize_t::wpkLenInByte

The byte length of wrapped private key for Rsa rep1,ECC,DSA and ECDSA case

Rsa rep1、ECC、DSA 和 ECDSA 情况下包装私钥的字节长度

CpaCyKptRsaWpkSizeRep2 CpaCyKptWpkSize_t::rsaWpkSizeRep2

CpaCyKptRsaWpkSizeRepCpaCyKptWpkSize_t::rsaWpkSizeRep

The byte length of wrapped private key for RSA rep2 case

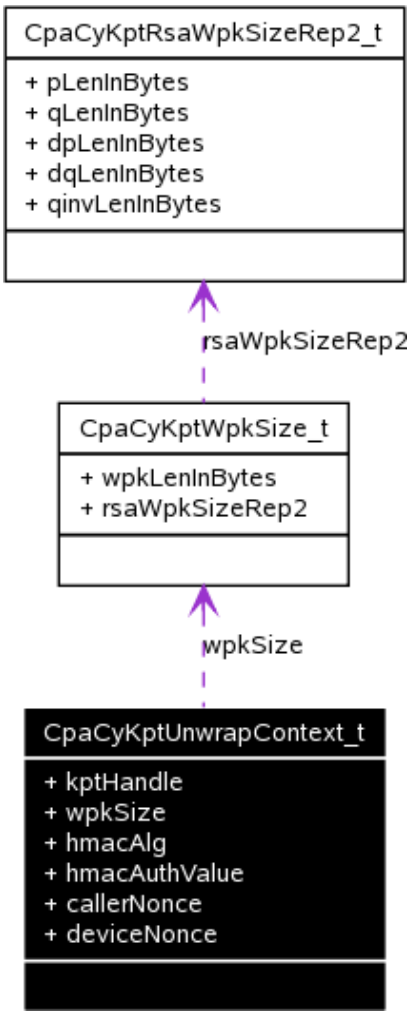
RSA rep2 情况下包装私钥的字节长度

22.7.7 CpaCyKptUnwrapContext_t Struct Reference

22.7.8 CpaCyKptUnwrapContext_t 结构引用

Collaboration diagram for CpaCyKptUnwrapContext_t:

CpaCyKptUnwrapContext_t 的协作图:



22.7.8.1 Detailed Description

22.7.8.2 详细描述

Structure of KPT unwrapping context.
KPT 展开上下文的结构。

This structure is a parameter of KPT crypto APIs, it contains data relating to KPT WPK unwrapping and HMAC authentication,application should complete those information in structure.
这个结构是 KPT 加密 API 的一个参数，它包含与 KPT WPK 展开和 HMAC 认证相关的数据，应用程序应该在结构中完成这些信息。

22.7.8.3 Data Fields
22.7.8.4 数据字段

- **CpaCyKptHandle kptHandle**

- CpaCyKptHandle kptHandle
- **CpaCyKptWpkSize wpkSize**
- CpaCyKptWpkSize wpkSize
- **CpaCyKptHMACType hmacAlg**
- CpaCyKptHMACType hmacAlg
- **Cpa8U hmacAuthValue** [CPA_CY_KPT_HMAC_LENGTH]
- Cpa8U hmacAuthValue [CPA _ CY _ KPT _ HMAC _长度]
- **Cpa8U callerNonce** [CPA_CY_KPT_CALLER_NONCE_LENGTH]
- Cpa8U callerNonce [CPA _ CY _ KPT _来电_现时_长度]
- **Cpa8U deviceNonce** [CPA_CY_KPT_DEVICE_NONCE_LENGTH]
- Cpa8U deviceNonce [CPA _ CY _ KPT _设备_随机数_长度]

22.7.8.5 Field Documentation

22.7.8.6 现场文件

CpaCyKptHandle CpaCyKptUnwrapContext_t::kptHandle

CpaCyKptHandlCpaCyKptUnwrapContext_t::kptHandl

This is application's unique handle that identifies its (symmetric) wrapping key

这是应用程序的唯一句柄，用于标识它的(对称)包装密钥

WPK's key size
WPK 的钥匙尺寸

HMAC algorithm used in HMAC authentication in KPT crypto service
HMAC 算法在 KPT 密码服务 HMAC 认证中的应用

HMAC authentication value input by the application in KPT crypto service;
KPT 加密服务中应用程序输入的 HMAC 身份验证值；

Caller(app) nonce generated by app in KPT crypto service
KPT 加密服务中的应用程序生成的调用方(应用程序)随机数

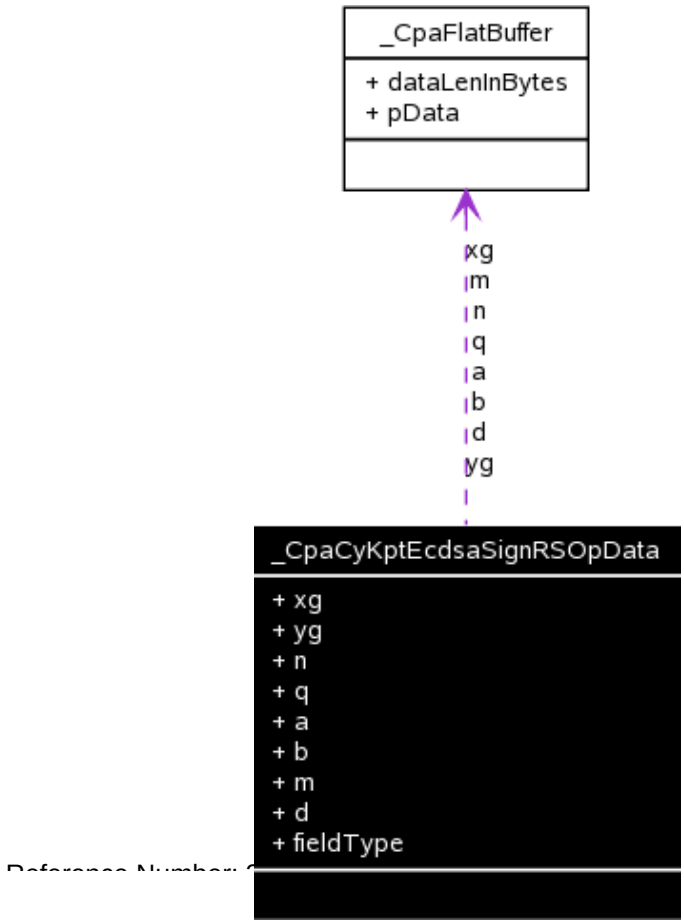
Device nonce generated by device in KPT crypto service
由 KPT 加密服务中的设备生成的设备随机数

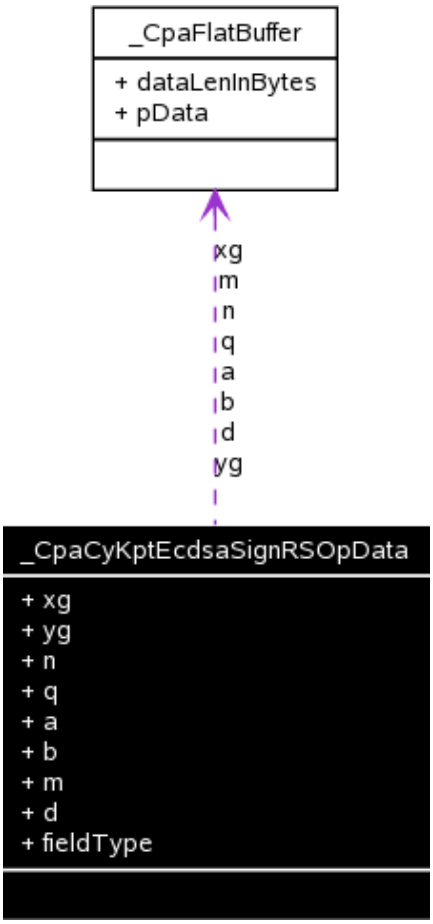
22.7.9 _CpaCyKptEcdsaSignRSOpData Struct Reference

22.7.10 _CpaCyKptEcdsaSignRSOpData 结构引用

Collaboration diagram for _CpaCyKptEcdsaSignRSOpData:

_CpaCyKptEcdsaSignRSOpData 的协作图:




_CpaFlatBuffer
+ dataLenInBytes
+ pData
xg
m
n
q
a
b
d
yg
_CpaCyKptEcdsaSignRSOpData
+ xg
+ yg
+ n
+ q
+ a
+ b
+ m
+ d
+ fieldType

22.7.10.1 Detailed Description

22.7.10.2 详细描述

KPTECDSA Sign R & S Operation Data.
KPTECDSA 签署 R & S 运营数据。

This structure contains the operation data for the cpaCyKptEcdsaSignRS function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.
此结构包含 cpaCyKptEcdsaSignRS 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

For optimal performance all data buffers SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据缓冲器都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.
该结构中的所有值都要求以最高有效字节优先，例如 a.pData[0] = MSB。

Note:
注意：

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyKptEcdsaSignRS function, and before it has been returned in the callback, undefined behavior will result.
如果客户端在将此结构中引用的内存提交给 cpaCyKptEcdsaSignRS 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

See also:
另请参见：

cpaCyEcdsaSignRS()
cpaCyEcdsaSignRS()

22.7.10.3 Data Fields
22.7.10.4 数据字段

- **CpaFlatBuffer xg**
- CpaFlatBuffer xg
- **CpaFlatBuffer yg**
- CpaFlatBuffer yg
- **CpaFlatBuffer n**
- CpaFlatBuffer n
- **CpaFlatBuffer q**
- CpaFlatBuffer q
- **CpaFlatBuffer a**
- CpaFlatBuffer a
- **CpaFlatBuffer b**
- CpaFlatBuffer b
- **CpaFlatBuffer m**

- CpaFlatBuffer m
- **CpaFlatBuffer d**
- CpaFlatBuffer d
- **CpaCyEcFieldType fieldType**
- CpaCyEcFieldType fieldType

22.7.10.5 Field Documentation

22.7.10.6 现场文件

x coordinate of base point G
基点 G 的 x 坐标

y coordinate of base point G
基点 G 的 y 坐标

order of the base point G, which shall be prime
应该是质数的基点 G 的阶

prime modulus or irreducible polynomial over GF(2^r)
GF(2^r 上的素数模或不可约多项式)

a elliptic curve coefficient
椭圆曲线系数

b elliptic curve coefficient
b 椭圆曲线系数

22.10 Define Documentation

22.11 定义文档

digest of the message to be signed
要签名的消息的摘要

private key
私人密钥

field type for the operation
操作的字段类型

22.8 Define Documentation

22.9 定义文档

Max length of initialization vector
初始化向量的最大长度

Defines the permitted max iv length in bytes that may be used in private key wrapping/unwrapping.For AEC-GCM,iv length is 12 bytes,for AES-CBC,iv length is 16 bytes.
定义可以在私钥包装/解包中使用的最大 iv 长度(以字节为单位)。对于 AEC-GCM，iv 长度为 12 字节，对于 AES-CBC，iv 长度为 16 字节。

See also:
另请参见:

cpaCyKptWrappingFormat
cpaCyKptWrappingFormat

Max length of HMAC value in HMAC authentication during KPT crypto service.
KPT 加密服务期间 HMAC 身份验证中 HMAC 值的最大长度。

Defines the permitted max HMAC value length in bytes that may be used to do HMAC verification in KPT crypto service.
定义允许的最大 HMAC 值长度(以字节为单位),可用于在 KPT 加密服务中进行 HMAC 验证。

See also:
另请参见:

cpaCyKptUnwrapContext
cpaCyKptUnwrapContext

Length of nonce generated by application in HMAC authentication during KPT crypto service.
KPT 加密服务期间 HMAC 身份验证中应用程序生成的随机数的长度。

Defines the caller nonce length in bytes that will be used to do HMAC authentication in KPT crypto service.
以字节为单位定义调用方随机数长度，该长度将用于在 KPT 加密服务中进行 HMAC 身份验证。

See also:
另请参见：
cpaCyKptUnwrapContext
cpaCyKptUnwrapContext

#define CPA_CY_KPT_DEVICE_NONCE_LENGTH

Length of nonce generated by QAT in HMAC authentication during KPT crypto service.
KPT 加密服务期间 HMAC 身份验证中 QAT 生成的 nonce 的长度。

Defines the device nonce length in bytes that will be used to do HMAC authentication in KPT crypto service.
定义将用于在 KPT 加密服务中进行 HMAC 身份验证的设备随机数长度(以字节为单位)。

See also:
另请参见：
cpaCyKptUnwrapContext
cpaCyKptUnwrapContext

22.10 Typedef Documentation

22.11 Typedef 文档

typedef Cpa64U CpaCyKptHandle

KPT wrapping key handle
KPT 包装钥匙把手

22.9 Typedef Documentation

22.10 Typedef 文档

Handle to a unique wrapping key in wrapping key table. Application creates it in KPT key transfer phase and maintains it for KPT Crypto service. For each KPT Crypto service API invocation,this handle will be used to get a SWK(Symmetric Wrapping Key) to unwrap WPK(Wrapped Private Key) before performing the requested crypto service.

包装关键字表中唯一包装关键字的句柄。应用程序在 KPT 密钥传输阶段创建它，并为 KPT 加密服务维护它。对于每个 KPT 加密服务 API 调用，此句柄将用于在执行请求的加密服务之前获取 SWK(对称包装密钥)来解开 WPK(包装的私钥)。

typedef enum **CpaCyKptWrappingKeyType_t CpaCyKptWrappingKeyType**

Cipher algorithms used to generate a wrapped private key (WPK) from the clear private key.

This enumeration lists supported cipher algorithms and modes.

用于从明文私钥生成包装私钥(WPK)的密码算法。此枚举列出了支持的密码算法和模式。

typedef enum **CpaCyKptHMACType_t CpaCyKptHMACType**

Hash algorithms used to generate WPK hash tag or used to do HMAC authentication in KPT crypto service.

This enumeration lists supported hash algorithms.

哈希算法用于生成 WPK 哈希标签或用于 KPT 加密服务中的 HMAC 认证。此枚举列出了支持的哈希算法。

typedef enum **CpaCyKptKeyManagementStatus_t CpaCyKptKeyManagementStatus**

Return Status

退货状态

This enumeration lists all the possible return status after completing KPT APIs.

此枚举列出了完成 KPT API 后所有可能的返回状态。

typedef enum **CpaCyKptKeySelectionFlags_t CpaCyKptKeySelectionFlags**

Key selection flag.

键选择标志。

This enumeration lists possible actions to be performed during cpaCyKptLoadKeys invocation.

此枚举列出了在调用 cpaCyKptLoadKeys 期间可能要执行的操作。

typedef enum **CpaCyKptKeyAction_t CpaCyKptKeyAction**

Key action.

关键行动。

PTT architecture support a "per-use" HMAC authorization for accessing and using key objects stored in PTT. This HMAC check is based on the use of running nonces shared between the application and PTT. To stay compatible with PTT's security protocol, QAT implements HMAC authorization protocol. This flag, set first time in cpaCyKptLoadKeys, will be used to determine whether HMAC authorization must be processed when QAT decrypts WPKs using SWKs.

PTT 架构支持用于访问和使用 PTT 中存储密钥对象的“每次使用”HMAC 授权。该 HMAC 检查基于应用和 PTT 之间共享的运行随机数的使用。为了与 PTT 的安全协议保持兼容，QAT 实现了 HMAC 授权协议。该标志第一次在 cpaCyKptLoadKeys 中设置，将用于确定当 QAT 使用 swk 解密 WPK 时，是否必须处理 HMAC 授权。

typedef struct **CpaCyKptWrappingFormat_t CpaCyKptWrappingFormat**

KPT wrapping format structure.
KPT 包装格式结构。

This structure defines wrapping format which is used to wrap clear private keys using a symmetric wrapping key.Application sets these parameters through the cpaCyKptLoadKeys calls.
该结构定义了包装格式，用于使用对称包装密钥包装明文私钥。应用程序通过调用 cpaCyKptLoadKeys 来设置这些参数。

typedef struct CpaCyKptRsaWpkSizeRep2_t CpaCyKptRsaWpkSizeRep2

RSA wrapped private key size structure For Representation 2.
表示 2 的 RSA 包装私钥大小结构。

This structure contains byte length of wrapped quintuple of p,q,dP,dQ and qInv which are required for the second representation of RSA private key. PKCS #1 V2.1 specification defines the second representation of the RSA private key,The quintuple of p, q, dP, dQ, and qInv are required for this representation.
此结构包含包装的五元组 p、q、dP、dQ 和 qInv 的字节长度，这是 RSA 私钥的第二种表示法所需要的。PKCS #1 V2.1 规范定义了 RSA 私钥的第二种表示，这种表示需要 p、q、dP、dQ 和 qInv 的五元组。

CpaCyRsaPrivateKeyRep2
CpaCyRsaPrivateKeyRep2

typedef union CpaCyKptWpkSize_t CpaCyKptWpkSize

Wrapped private key size union.
包装私钥大小联合。

A wrapped private key size union, either wrapped quintuple of RSA representation 2 private key,or byte length of wrapped ECC/RSA Rep1/DSA/ ECDSA private key.
包装的私钥大小联合，或者是 RSA 表示 2 私钥的包装五元组，或者是包装的 ECC/RSA Rep1/DSA/ ECDSA 私钥的字节长度。

typedef struct **CpaCyKptUnwrapContext** *CpaCyKptUnwrapContext*

Structure of KPT unwrapping context.
KPT 展开上下文的结构。

This structure is a parameter of KPT crypto APIs, it contains data relating to KPT WPK unwrapping and HMAC authentication,application should complete those information in structure.
这个结构是 KPT 加密 API 的一个参数，它包含与 KPT WPK 展开和 HMAC 认证相关的数据，应用程序应该在结构中完成这些信息。

typedef struct **CpaCyKptEcdsaSignRSOpData** *CpaCyKptEcdsaSignRSOpData*

KPTECDSA Sign R & S Operation Data.
KPTECDSA 签署 R & S 运营数据。

This structure contains the operation data for the cpaCyKptEcdsaSignRS function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.
此结构包含 cpaCyKptEcdsaSignRS 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

For optimal performance all data buffers SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据缓冲区都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.
该结构中的所有值都要求以最高有效字节优先，例如 a.pData[0] = MSB。

Note:
注意:

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyKptEcdsaSignRS function, and before it has been returned in the callback, undefined behavior will result.
如果客户端在将此结构中引用的内存提交给 cpaCyKptEcdsaSignRS 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

See also:
另请参见:

cpaCyEcdsaSignRS()
cpaCyEcdsaSignRS()

22.11 Enumeration Type Documentation

22.12 枚举类型文档

enum **CpaCyKptWrappingKeyType** *t*

Cipher algorithms used to generate a wrapped private key (WPK) from the clear private key.

This enumeration lists supported cipher algorithms and modes.

用于从明文私钥生成包装私钥(WPK)的密码算法。此枚举列出了支持的密码算法和模式。

enum **CpaCyKptHMACType_t**

Hash algorithms used to generate WPK hash tag or used to do HMAC authentication in KPT crypto service.

This enumeration lists supported hash algorithms.

CpaCyKptHMACType_t

哈希算法用于生成 WPK 哈希标签或用于 KPT 加密服务中的 HMAC 认证。此枚举列出了支持的哈希算法。

Enumerator:
枚举器:

CPA_CY_KPT_HMAC_TYPE_NULL No HMAC required
CPA_CY_KPT_HMAC_TYPE_NULL 不需要 HMAC

Return Status enum **CpaCyKptKeyManagementStatus_t**
退货状态

CpaCyKptKeyManagementStatus_t

This enumeration lists all the possible return status after completing KPT APIs.
此枚举列出了完成 KPT API 后所有可能的返回状态。

Enumerator:
枚举器:

CPA_CY_KPT_SUCCESS *CPA _ CY _ KPT _成功一般成功*	Generic success status for all KPT wrapping key handling functions 所有 KPT 包装按键处理功能的状态
CPA_CY_KPT_REGISTER_HANDLE_FAIL_RETRY *CPA _ CY _ KPT _ REGISTER _ HANDLE _ FAIL _ RETRY WKT 正忙，请稍后重试*	WKT is busy, retry after some time

CPA_CY_KPT_REGISTER_HANDLE_FAIL_DUPLICATE	Handle is already present in WKT; this is attempt at duplication *WKT 中已经存在 CPA _ CY _ KPT _ REGISTER _ HANDLE _ FAIL _ DUPLICATE 句柄；这是试图复制*
CPA_CY_KPT_LOAD_KEYS_FAIL_INVALID_HANDLE	LoadKey call does not provide a handle that was previously registered. Either application error, or malicious application. Reject request to load the key. *CPA _ CY _ KPT _ LOAD _ KEYS _ FAIL _ INVALID _ HANDLE LOAD key 调用不提供以前注册的句柄。要么是应用错误，要么是恶意应用。拒绝加载密钥的请求。*
CPA_CY_KPT_REGISTER_HANDLE_FAIL_WKT_FULL	Failed to register wrapping key as WKT is full *由于 WKT 已满，CPA _ CY _ KPT _ REGISTER _ HANDLE _ FAIL _ WKT _ FULL 无法注册回绕密钥*
CPA_CY_KPT_WKT_ENTRY_NOT_FOUND	Unable to find SWK entry by handle *CPA_CY_KPT_WKT_ENTRY_NOT_FOUND 无法通过句柄找到 SWK 条目*
CPA_CY_KPT_REGISTER_HANDLE_FAIL_INSTANCE_QUOTA_EXCEEDED	This application has opened too many WKT entries. A Quota is enforced to prevent DoS attacks *CPA _ CY _ KPT _ REGISTER _ HANDLE _ FAIL _ INSTANCE _ QUOTA _ exceed 此应用程序打开了太多的 WKT 条目。强制实施配额以防止 DoS 攻击*
CPA_CY_KPT_LOADKEYS_FAIL_CHECKSUM_ERROR	Checksum error in key loading *CPA _ CY _ KPT _ load keys _ FAIL _ CHECKSUM _ ERROR 加载密钥时出现校验和错误*
CPA_CY_KPT_LOADKEYS_FAIL_HANDLE_NOT_REGISTERED	Key is not registered in key loading *CPA _ CY _ KPT _ load keys _ FAIL _ HANDLE _ NOT _ REGISTERED Key 不是在密钥加载中注册*
CPA_CY_KPT_LOADKEYS_FAIL_POSSIBLE_DOS_ATTACK	Possible Dos attack happened in key loading *CPA _ CY _ KPT _ load keys _ FAIL _ POSSIBLE _ DOS _ ATTACK 可能的 DOS 密钥加载时发生攻击*
CPA_CY_KPT_LOADKEYS_FAIL_INVALID_AC_SEND_HANDLE	Invalid key handle

CPA_CY_KPT_load keys_FAIL_INVALID_AC_SEND_HANDLE 无效密钥句柄

	got from PTT 从 PTT 获取
CPA_CY_KPT_LOADKEYS_FAIL_INVALID_DATA_OBJ	Invalid data object got from PTT

CPA_CY_KPT_load keys_FAIL_INVALID_DATA_OBJ 从 PTT 获取无效数据对象

enum **CpaCyKptKeySelectionFlags_t**

列举型别 CpaCyKptKeySelectionFlags_

Key selection flag.
键选择标志。

This enumeration lists possible actions to be performed during cpaCyKptLoadKeys invocation.
此枚举列出了在调用 cpaCyKptLoadKeys 期间可能要执行的操作。

Enumerator:
枚举器:

CPA_CY_KPT_SWK	Symmetric wrapping key,only a SWK will be loaded from PTT to QAT
CPA_CY_KPT_SWK 对称回绕密钥，只有一个 SWK 会从 PTT 加载到 QAT	
CPA_CY_KPT_WPK	Wrapped private key, a data blob including SWK and CPK will be loaded from PTT to QAT, and WPK will be return to application.
CPA_CY_KPT_WPK 包装私钥，包括 SWK 和 CPK 的数据 blob 将从 PTT 加载到 QAT，WPK 将返回给应用。	
CPA_CY_KPT_OPAQUE_DATA	Opaque data,a opaque data will be loaded from PTT to QAT QAT
CPA_CY_KPT_OPAQUE_DATA 不透明数据，一个不透明数据将从 PTT 加载到	
CPA_CY_KPT_HMAC_AUTH_PARAMS	HMAC auth params,HMAC auth params will be loaded
CPA_CY_KPT_HMAC_验证参数 HMAC 验证参数，HMAC 验证参数将被加载	

from PTT to QAT

从 PTT 到 QAT

CPA_CY_KPT_RN_SEED DRBG seed,A rondom data generated by PTT will be loaded from PTT to QAT

CPA_CY_KPT_RN_SEED DRBG seed，PTT 生成的一个随机数据将从 PTT 加载到 QAT

enum **CpaCyKptKeyAction_t**

列举型别 CpaCyKptKeyAction_

Key action.

关键行动。

PTT architecture support a "per-use" HMAC authorization for accessing and using key objects stored in PTT. This HMAC check is based on the use of running nonces shared between the application and PTT. To stay compatible with PTT's security protocol, QAT implements HMAC authorization protocol. This flag, set first time in cpaCyKptLoadKeys, will be used to determine whether HMAC authorization must be processed when QAT decrypts WPKs using SWKs.

PTT 架构支持用于访问和使用 PTT 中存储密钥对象的“每次使用”HMAC 授权。该 HMAC 检查基于应用和 PTT 之间共享的运行随机数的使用。为了与 PTT 的安全协议保持兼容，QAT 实现了 HMAC 授权协议。该标志第一次在 cpaCyKptLoadKeys 中设置，将用于确定当 QAT 使用 swk 解密 WPK 时，是否必须处理 HMAC 授权。

Enumerator:

枚举器:

CPA_CY_KPT_NO_HMAC_AUTH_CHECK Do not need HMAC authentication check in KPT Crypto service

*CPA_CY_KPT_NO_HMAC_AUTH_CHECK 在 KPT 不需要 HMAC 认证检查*加密服务

CPA_CY_KPT_HMAC_AUTH_CHECK Need HMAC authentication check in KPT Crypto service

*KPT 加密需要 HMAC 认证检查*服务

22.13 Function Documentation

22.14 功能文档

```
cpaCyKptRegisterKeyHandle (                                    instanceHandle
cpaCyKptRegisterKeyHandle (                                    instanceHandle,
                                keyHandle,
                                pKptStatus
                                *
                            )
```

Perform KPT key handle register function.

执行 KPT 键句柄寄存器功能。

Used for loading an application's wrapping key from PTT to QAT. An application first precomputes/initializes a 64 bit handle value using CPU based RDRAND instruction or other means and passes it to QAT. This will

signal to QAT that a KPT key transfer operation is about to begin
用于将应用程序的包装密钥从 PTT 加载到 QAT。应用程序首先使用基于 CPU 的 RDRAND 指令或其他手段预计算/初始化 64 位句柄值，并将其传递给 QAT。这将向 QAT 发信号通知 KPT 密钥传送操作即将开始

Context:
背景：

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.
这是一个同步功能，它可以休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设：

None
没有人

Side-Effects:
副作用：

None
没有人

Blocking:
阻止：

This function is synchronous and blocking.
这个函数是同步的和阻塞的。

Reentrant:
可重入：

No
不

Thread-safe:
线程安全：

Yes
是

Parameters:
参数：

[in] *instanceHandle* QAT service instance handle.
[in] instanceHandle QAT 服务实例句柄。

[in] *keyHandle* A 64-bit handle value

[in]key handle 64 位句柄值

[out] *pKptStatus* One of the status codes denoted in the enumerate type of cpaCyKptKeyManagementStatus

[out] pKptStatus 在 cpaCyKptKeyManagementStatus 的枚举类型中指示的状态代码之一

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_INVALID_PARAM Invalid parameter passed in.

CPA_STATUS_RESOURCE Error related to system resources.

CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。

CPA_STATUS_RESTARTING API implementation is restarting. Resubmit the request.

CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。

Precondition:

前提条件:

Component has been initialized.

组件已初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

This function operates in a synchronous manner and no asynchronous callback will be generated.

该函数以同步方式运行，不会生成异步回调。

See also:

另请参见:

None

没有人

```
cpaCyKptLoadKeys (                                                    instanceHandle, keyHandle, pKptWrappingFo
cpaCyKptLoadKeys (                                                    instanceHandle, keyHandle, pKptWrappingFo
 *
                                 *pOutputData,
                                 * pKptStatus
                               )
```

Perform KPT key loading function.

执行 KPT 键加载功能。

This function is invoked by QAT application after instructing PTT to send its wrapping key to QAT. After PTT returns a TPM_SUCCESS, the wrapping key structure is placed in QAT. The Application completes the 3-way handshake by invoking this API and requesting QAT to store the wrapping key, along with its handle.
在指示 PTT 向 QAT 发送其包装密钥后，QAT 应用程序调用该函数。在 PTT 返回 TPM_SUCCESS 之后，包装密钥结构被放置在 QAT 中。应用程序通过调用这个 API 并请求 QAT 存储包装密钥及其句柄来完成 3 次握手。

Context:
背景：

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.
这是一个同步功能，它可以休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设：

None
没有人

Side-Effects:
副作用：

None
没有人

Blocking:
阻止：

This function is synchronous and blocking.
这个函数是同步的和阻塞的。

Reentrant:
可重入：

No
不

Thread-safe:
线程安全：

Yes

是

Parameters:
参数:

[in] *instanceHandle* QAT service instance handle.
[in] instanceHandle QAT 服务实例句柄。
[in] *keyHandle* A 64-bit handle value
[in]key handle 64 位元的控制代码值
[in] *keySelFlag* Flag to indicate which kind of mode (SWK or WPK) should be
[in] keySelFlag 旗标，指出应该是哪种模式(SWK 或 WPK)
loaded.
上膛了。
[in] *keyAction* Whether HAMC authentication is needed
[in]关键操作是否需要 HAMC 身份验证
[in] *pKptWrappingFormat* Pointer to CpaCyKptWrappingFormat whose fields will be written to
[in]指向 CpaCyKptWrappingFormat 的 pKptWrappingFormat 指标，其栏位将会写入
WKT.
WKT。
[out] *pOutputData* FlatBuffer pointer, which contains the wrapped private key structure
[out] pOutputData FlatBuffer 指标，包含包装的私密金钥结构
used by application.
由应用程序使用。
[out] *pKptStatus* One of the status codes denoted in the enumerate type
[out] pKptStatus 枚举类型中指示的状态代码之一
CpaCyKptKeyManagementStatus
CpaCyKptKeyManagementStatus

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。
CPA_STATUS_FAIL Function failed.
CPA_STATUS_INVALID_PARAM Invalid parameter passed in.
CPA_STATUS_RESOURCE Error related to system resources.
CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。
CPA_STATUS_RESTARTING API implementation is restarting. Resubmit the request.
CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。

Precondition:
前提条件:

Component has been initialized.
组件已初始化。

Postcondition:
后置条件:

None
没有人

Note:

注意:

None
没有人

See also:

另请参见:

None
没有人

```
cpaCyKptDeleteKey (                                          instanceHandle
cpaCyKptDeleteKey (                                          instanceHandle,
                            keyHandle,
                            * pkptstatus
                          )
```

Perform KPT delete keys function according to key handle
根据按键处理执行 KPT 删除按键功能

Before closing a QAT session(instance), an application that has previously stored its wrapping key in QAT using the KPT framework executes this call to delete its wrapping key in QAT.
在关闭 QAT 会话(实例)之前，之前使用 KPT 框架将其包装密钥存储在 QAT 中的应用程序执行此调用，以删除其在 QAT 中的包装密钥。

Context:
背景:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.
这是一个同步功能，它可以休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

This function is synchronous and blocking.
这个函数是同步的和阻塞的。

Reentrant:

可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

[in] *instanceHandle* QAT service instance handle.
[in] instanceHandle QAT 服务实例句柄。
[in] *keyHandle* A 64-bit handle value
[in]key handle 64 位句柄值
[out] *pkptstatus* One of the status codes denoted in the enumerate type
[out] pkptstatus 枚举类型中指示的状态代码之一
CpaCyKptKeyManagementStatus
CpaCyKptKeyManagementStatus

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。
CPA_STATUS_FAIL Function failed.
CPA_STATUS_INVALID_PARAM Invalid parameter passed in.
CPA_STATUS_RESOURCE Error related to system resources.
CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。
CPA_STATUS_RESTARTING API implementation is restarting. Resubmit the request.
CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。

Precondition:
前提条件:

Component has been initialized.
组件已初始化。

Postcondition:
后置条件:

None
没有人

Note:

注意:

None
没有人

See also:

另请参见:

None
没有人

CpaStatus cpaCyKptRsaDecrypt (const CpaInstanceHandle *instanceHandle*,
CpaStatus cpaCyKptRsaDecrypt (const CpaInstanceHandle *instanceHandle*,
const CpaCyGenFlatBufCbFunc
const CpaCyGenFlatBufCbFunc
pRsaDecryptCb, void * *pCallbackTag*,
const CpaCyRsaDecryptOpData * *pDecryptOpData*,
CpaFlatBuffer * *pOutputData*
CpaFlatBuffer *

Perform KPT mode RSA decrypt primitive operation on the input data.
对输入数据执行 KPT 模式 RSA 解密原语操作。

This function is variant of cpaCyRsaDecrypt, which will perform an RSA decryption primitive operation on the input data using the specified RSA private key which are encrypted. As the RSA decryption primitive and signing primitive operations are mathematically identical this function may also be used to perform an RSA signing primitive operation.
此函数是 cpaCyRsaDecrypt 的变体，它将使用加密的指定 RSA 私钥对输入数据执行 RSA 解密原语操作。由于 RSA 解密原语和签名原语操作在数学上是相同的，因此该函数也可以用于执行 RSA 签名原语操作。

Context:
背景:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:

阻止:

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

[in] *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。

[in] *pRsaDecryptCb* Pointer to callback function to be invoked when the operation is complete. If this is set to a NULL value the function will operate synchronously.
[in]当作业为时，要叫用之回呼函式的 pRsaDecryptCb 指标
完成。如果设置为空值，函数将同步运行。

[in] *pCallbackTag* Opaque User Data for this specific call. Will be returned unchanged in the callback.
[in] pCallbackTag 此特定调用的不透明用户数据。将在回调中不变地返回。

[in] *pDecryptOpData* Structure containing all the data needed to perform the RSA decrypt operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
[in] pDecryptOpData 结构，包含执行 RSA 解密所需的所有资料
操作。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

[out] *pOutputData* Pointer to structure into which the result of the RSA decryption primitive is written. The client MUST allocate this memory. The data pointed to is an integer in big-endian order. The value will be between 0 and the modulus n - 1. On invocation the callback function will contain this parameter in the pOut parameter.
[out] pOutputData 指向 RSA 解密结果所在的结构的指针
原语是写的。客户端必须分配这个内存。指向的数据是以大端顺序排列的整数。该值将在 0 和模数 n - 1 之间。在调用时，回调函数将在 pOut 参数中包含这个参数。

[in] *pKptUnwrapContext* Pointer of structure into which the content of KptUnwrapContext is kept,The client MUST allocate this memory and copy structure KptUnwrapContext into this flat buffer.
[in]结构的 pKptUnwrapContext 指标，KptUnwrapContext 的内容会放入其中
保持，客户端必须分配这个内存并将结构 KptUnwrapContext 复制到这个平面缓冲区中。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。
CPA_STATUS_FAIL Function failed.
CPA_STATUS_FAIL 函数失败。
CPA_STATUS_RETRY Resubmit the request.
CPA_STATUS_INVALID_PARAM Invalid parameter passed in.
CPA_STATUS_RESOURCE Error related to system resources.
CPA_STATUS_RESTARTING API implementation is restarting.Resubmit the request.
CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.
该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:
后置条件:

None
没有人

Note:

注意:

By virtue of invoking cpaSyKptRsaDecrypt, the implementation understands that pDecryptOpData contains an encrypted private key that requires unwrapping. KptUnwrapContext contains an 'KptHandle' field that points to the unwrapping key in the WKT. When pRsaDecryptCb is non-NULL an asynchronous callback is generated in response to this function call. Any errors generated during processing are reported as part of the callback status code. For optimal performance, data pointers SHOULD be 8-byte aligned. In KPT release,private key field in CpaCyRsaDecryptOpData is a concatenation of cipher text and hash tag. For optimal performance, data pointers SHOULD be 8-byte aligned. 通过调用 cpaSyKptRsaDecrypt，该实现知道 pDecryptOpData 包含需要解包的加密私钥。KptUnwrapContext 包含一个指向 WKT 中的展开密钥的“KptHandle”字段。当 pRsaDecryptCb 为非 NULL 时，会生成一个异步回调来响应此函数调用。处理过程中产生的任何错误都会作为回调状态代码的一部分进行报告。为了获得最佳性能，数据指针应该 8 字节对齐。在 KPT 版本中，CpaCyRsaDecryptOpData 中的私钥字段是密文和散列标签的串联。为了获得最佳性能，数据指针应该 8 字节对齐。	

See also:

另请参见：

CpaCyRsaDecryptOpData, **CpaCyGenFlatBufCbFunc**, **cpaCyRsaGenKey()**, **cpaCyRsaEncrypt()**

CpaCyRsaDecryptOpData, CpaCyGenFlatBufCbFunccpaCyRsaGenKey () cpaCyRsaEncrypt()

```
cpaCyKptEcdsaSignRS ( const                                    instanceHandle, pCb, pCallbackTag,
cpaCyKptEcdsaSignRS (  const                                   instanceHandle, pCb, pCallbackTag
const
void *
                                  const * pOpData,
                                  *pSignStatus,
                                  *pR,
                                  *pS,
                                  *pKptUnwrapContext
                                )
```

Generate ECDSA Signature R & S.
生成 ECDSA 签名 R & S。

This function is a varient of cpaCyEcdsaSignRS, it generates ECDSA Signature R & S as per ANSI X9.62 2005 section 7.3.
此函数是 cpaCyEcdsaSignRS 的变体，它根据 ANSI X9.62 2005 第 7.3 节生成 ECDSA 签名 R & S。

Context:
背景:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入:

No
不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

[in] *instanceHandle* Instance handle.

[in] instanceHandle 执行个体控制代码。

[in] *pCb* Callback function pointer. If this is set to a NULL value the function

[in] pCb 回调函数指针。如果设置为空值，函数

will operate synchronously.

将同步运行。

[in] *pCallbackTag* User-supplied value to help identify request.

[in] pCallbackTag 使用者提供的值，可协助识别要求。

[in] *pOpData* Structure containing all the data needed to perform the operation.

[in] pOpData 结构，包含执行作业所需的所有资料。

The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.

客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

[out] *pSignStatus* In synchronous mode, the multiply output is valid (CPA_TRUE) or the

[out] pSignStatus 在同步模式下，乘法输出有效(CPA_TRUE)或

output is invalid (CPA_FALSE).

输出无效(CPA_FALSE)。

[out] *pR* ECDSA message signature r.

[out] pR ECDSA 消息签名 r。

[out] *pS* ECDSA message signature s.

[out] pS ECDSA 消息签名。

[in] *pKptUnwrapContext* Pointer of structure into which the content of KptUnwrapContext is

[in]结构的 pKptUnwrapContext 指标，KptUnwrapContext 的内容会放入其中

kept,The client MUST allocate this memory and copy structure KptUnwrapContext into this flat buffer.

保持，客户端必须分配这个内存并将结构 KptUnwrapContext 复制到这个平面缓冲区中。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_FAIL 函数失败。

CPA_STATUS_RETRY Resubmit the request.

CPA_STATUS_INVALID_PARAM Invalid parameter passed in.

CPA_STATUS_RESOURCE Error related to system resources.

CPA_STATUS_RESTARTING API implementation is restarting. Resubmit the request.

CPA_STATUS_UNSUPPORTED Function is not supported.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.

该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

By virtue of invoking the cpaCyKptEcdsaSignRS, the implementation understands CpaCyEcdsaSignRSOpData contains an encrypted private key that requires unwrapping. KptUnwrapContext contains an 'KptHandle' field that points to the unwrapping key in the WKT. When pCb is non-NULL an asynchronous callback of type CpaCyEcdsaSignRSCbFunc generated in response to this function call. In KPT release,private key field in CpaCyEcdsaSignRSOpData is a concatenation of cipher text and hash tag. 通过调用 cpaCyKptEcdsaSignRS，该实现知道 CpaCyEcdsaSignRSOpData 包含需要解包的加密私钥。KptUnwrapContext 包含一个指向 WKT 中的展开密钥的“KptHandle”字段。当 pCb 为非空时，会生成一个类型为 CpaCyEcdsaSignRSCbFunc 的异步回调来响应此函数调用。在 KPT 版本中，CpaCyEcdsaSignRSOpData 中的私钥字段是密文和散列标签的串联。	

See also:

另请参见:

None
没有人

```
cpaCyKptDsaSignS ( const                                    instanceHandle, pCb,
cpaCyKptDsaSignS (  const                                   instanceHandle, pCb,
const
                            void *pCallbackTag,
                            const * pOpData,
                            *pProtocolStatus,
                            *pS,
                            *pKptUnwrapContext
                          )
```

This function is varient of cpaCyDsaSignS,which generate DSA S Signature.
这个函数是 cpaCyDsaSignS 的变体，它生成 DSA 的签名。

This function generates the DSA S signature as described in FIPS 186-3 Section 4.6: s = (k^-1(z + xr)) mod q
该函数生成 DSA 的签名，如 FIPS 186-3 第 4.6 节所述:s = (kˆ-1(z + xr)) mod q

Here, z = the leftmost min(N, outlen) bits of Hash(M). This function does not perform the SHA digest; z is computed by the caller and passed as a parameter in the pOpData field.
这里，z = Hash(M)最左边的 min(N，outlen)位。此函数不执行 SHA 摘要；z 由调用者计算，并作为参数在 pOpData 字段中传递。

The protocol status, returned in the callback function as parameter protocolStatus (or, in the case of synchronous invocation, in the parameter *pProtocolStatus) is used to indicate whether the value s == 0.
在回调函数中作为参数 protocol status(或者，在同步调用的情况下，在参数*pProtocolStatus 中)返回的协议状态用于指示值 s == 0。

Specifically, (protocolStatus == CPA_TRUE) means s != 0, while (protocolStatus == CPA_FALSE) means s == 0.
具体来说，(protocolStatus == CPA_TRUE)的意思是 s! = 0，而(protocolStatus == CPA_FALSE)表示 s == 0.

If signature r has been generated in advance, then this function can be used to generate the signature s once the message becomes available.
如果已经预先生成了签名 r，那么一旦消息变得可用，就可以使用该函数来生成签名 s。

Context:
背景：

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:

假设:

None
没有人

Side-Effects:

副作用:

None
没有人

Blocking:

阻止:

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:

可重入:

No
不

Thread-safe:

线程安全:

Yes
是

Parameters:

参数:

[in] *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。

[in] *pCb* Callback function pointer. If this is set to a NULL value the function
[in] pCb 回调函数指针。如果设置为空值，函数
will operate synchronously.
将同步运行。

[in] *pCallbackTag* User-supplied value to help identify request.
[in] pCallbackTag 使用者提供的值，可协助识别要求。

[in] *pOpData* Structure containing all the data needed to perform the operation.
[in] pOpData 结构，包含执行作业所需的所有资料。
The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

[out] *pProtocolStatus* The result passes/fails the DSA protocol related checks.
[out] pProtocolStatus 结果通过/未通过 DSA 协议相关检查。

[out] *pS* DSA message signature s. On invocation the callback function will
[out] pS DSA 消息签名。在调用回调函数时，它将
contain this parameter in the pOut parameter.
在 pOut 参数中包含此参数。

[in] *pKptUnwrapContext* Pointer of structure into which the content of KptUnwrapContext is
[in]结构的 pKptUnwrapContext 指标，KptUnwrapContext 的内容会放入其中

kept,The client MUST allocate this memory and copy structure KptUnwrapContext into this flat buffer.
保持，客户端必须分配这个内存并将结构 KptUnwrapContext 复制到这个平面缓冲区中。

Return values:

返回值：

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。	
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_FAIL 函数失败。	
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件：

The component has been initialized via cpaCyStartInstance function.
该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:

后置条件：

None
没有人

Note:

注意：

When pCb	is non-NULL an asynchronous callback of type CpaCyDsaSSignCbFunc is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned. 当 pCb 为非空时，会生成一个 CpaCyDsaSSignCbFunc 类型的异步回调来响应此函数调用。为了获得最佳性能，数据指针应该 8 字节对齐。

By virtue of invoking cpaCyKptDsaSignS, the implementation understands CpaCyDsaSSignOpData contains an encrypted private key that requires unwrapping. KptUnwrapContext contains an 'KptHandle' field that points to the unwrapping key in the WKT. In KPT,private key field in CpaCyDsaSSignOpData is a concatenation of cipher text and hash tag. For optimal performance, data pointers SHOULD be 8-byte aligned.

通过调用 cpaCyKptDsaSignS，该实现知道 CpaCyDsaSSignOpData 包含一个需要解包的加密私钥。KptUnwrapContext 包含一个指向 WKT 中的展开密钥的“KptHandle”字段。在 KPT 中，CpaCyDsaSSignOpData 中的私钥字段是密文和散列标签的串联。为了获得最佳性能，数据指针应该 8 字节对齐。

See also:

另请参见:

CpaCyDsaSSignOpData, **CpaCyDsaGenCbFunc**, **cpaCyDsaSignR()**, **cpaCyDsaSignRS()**
CpaCyDsaSSignOpData,CpaCyDsaGenCbFunccpaCyDsaSignR()cpaCyDsaSignRS()

```
cpaCyKptDsaSignRS  (const                                    instanceHandle, pCb, pCallbackTag, pOpData,
cpaCyKptDsaSignRS   (const                                   instanceHandle, pCb, pCallbackTag, pOpData,
const  void *                                                pProtocolStatus, pR,
const * * * * *                                              pS, pKptUnwrapContext
                                )
```

This function is a varient of cpaCyDsaSignRS,which generate DSA R and S Signature
这个函数是 cpaCyDsaSignRS 的变体，它生成 DSA R 和 S 签名

This function generates the DSA R and S signatures as described in FIPS 186-3 Section 4.6:

r = (g^k mod p) mod q s = (k^-1(z + xr)) mod q

此函数生成 DSA R 和 s 签名，如 FIPS 186-3 第 4.6 节所述:r =(gˆk mod p)modq s =(kˆ-

1(z+xr))modq

Here, z = the leftmost min(N, outlen) bits of Hash(M). This function does not perform the SHA digest; z is computed by the caller and passed as a parameter in the pOpData field.
这里，z = Hash(M)最左边的 min(N，outlen)位。此函数不执行 SHA 摘要；z 由调用者计算，并作为参数在 pOpData 字段中传递。

The protocol status, returned in the callback function as parameter protocolStatus (or, in the case of synchronous invocation, in the parameter *pProtocolStatus) is used to indicate whether either of the values r or s are zero.
在回调函数中作为参数 protocol status(或者，在同步调用的情况下，在参数*pProtocolStatus 中)返回的协议状态用于指示值 r 或 s 是否为零。

Specifically, (protocolStatus == CPA_TRUE) means neither is zero (i.e. (r != 0) && (s != 0)), while (protocolStatus == CPA_FALSE) means that at least one of r or s is zero (i.e. (r == 0) || (s == 0)).
具体来说，(protocolStatus == CPA_TRUE)意味着两者都不为零(即(r！= 0) && (s！= 0))，而(protocolStatus == CPA_FALSE)表示 r 或 s 中至少有一个为零(即(r == 0) || (s == 0))。

Context:
背景:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设：

None
没有人

Side-Effects:
副作用：

None
没有人

Blocking:
阻止：

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入：

No
不

Thread-safe:
线程安全：

Yes
是

Parameters:

[in]	*instanceHandle*	Instance handle.
[in]	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
[in]	*pCallbackTag*	User-supplied value to help identify request.

参数：

［在］	*instanceHandle*	实例句柄。
［在］	*印刷电路板*	回调函数指针。如果设置为空值，函数将同步运行。
［在］	*pCallbackTag*	用户提供的帮助识别请求的值。

[in] *pOpData* Structure containing all the data needed to perform the operation.

[in] pOpData 结构，包含执行作业所需的所有资料。

The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

[out] *pProtocolStatus* The result passes/fails the DSA protocol related checks.

[out] pProtocolStatus 结果通过/未通过 DSA 协议相关检查。

[out] *pR* DSA message signature r.

[out] pR DSA 消息签名。

[out] *pS* DSA message signature s.

[out] pS DSA 消息签名。

[in] *pKptUnwrapContext* Pointer of structure into which the content of KptUnwrapContext is

[in]结构的 pKptUnwrapContext 指标，KptUnwrapContext 的内容会放入其中

kept,The client MUST allocate this memory and copy structure KptUnwrapContext into this flat buffer.
保持，客户端必须分配这个内存并将结构 KptUnwrapContext 复制到这个平面缓冲区中。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_FAIL 函数失败。

CPA_STATUS_RETRY Resubmit the request.

CPA_STATUS_INVALID_PARAM Invalid parameter passed in.

CPA_STATUS_RESOURCE Error related to system resources.

CPA_STATUS_RESTARTING API implementation is restarting. Resubmit the request.

CPA_STATUS_UNSUPPORTED Function is not supported.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.
该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:

后置条件:

None
没有人

Note:

注意:

When pCb is non-NULL an asynchronous callback of type CpaCyDsaRSSignCbFunc is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned. By virtue of invoking CyKptDsaSignRS, the implementation understands CpaCyDsaRSSignOpData contains an enrypted private key that requires unwrapping. 当 pCb 为非空时，会生成一个 CpaCyDsaRSSignCbFunc 类型的异步回调来响应此函数调用。为了获得最佳性能，数据指针应该 8 字节对齐。通过调用 CyKptDsaSignRS，该实现知道 CpaCyDsaRSSignOpData 包含一个需要解包的加密私钥。	KptUnwrapContext contains an 'KptHandle' field that points to the unwrapping key in the WKT. In KPT,private key field in CpaCyDsaRSSignOpData is a concatenation of cipher text and hash tag. For optimal performance, data pointers SHOULD be 8-byte aligned. KptUnwrapContext 包含一个指向 WKT 中的展开密钥的“KptHandle”字段。在 KPT 中，CpaCyDsaRSSignOpData 中私钥字段是密文和散列标签的串联。为了获得最佳性能，数据指针应该 8 字节对齐。

See also:

另请参见:

CpaCyDsaRSSignOpData, **CpaCyDsaRSSignCbFunc**, **cpaCyDsaSignR()**, **cpaCyDsaSignS()**
CpaCyDsaRSSignOpData, CpaCyDsaRSSignCbFunccpaCyDsaSignR() cpaCyDsaSignS()