Length of source buffer, or **CPA_DP_BUFLIST**.
源缓冲区的长度，或 CPA_DP_BUFLIST

Physical address of the destination buffer on which to operate. This is either:
要操作的目标缓冲区的物理地址。这要么是:

- The location of the data, of length srcBufferLen; or,
- 数据的位置，长度为 srcBufferLen 或者，
- If srcBufferLen has the special value **CPA_DP_BUFLIST**, then srcBuffer contains the location where a **CpaPhysBufferList** is stored. In this case, the CpaPhysBufferList MUST be aligned on an 8-byte boundary.
- 如果 srcBufferLen 具有特殊值 CPA_DP_BUFLISTCpaPhysBufferList

For "in-place" operation, the dstBuffer may be identical to the srcBuffer.
对于“就地”操作，dstBuffer 可以与 srcBuffer 相同。

Length of destination buffer, or **CPA_DP_BUFLIST**.
目标缓冲区的长度，或 CPA_DP_BUFLIST

Physical address of this data structure
该数据结构的物理地址

Pointer to (and therefore, the virtual address of) the IV field above. Needed here because the driver in some cases writes to this field, in addition to sending it to the accelerator.
指向上述 IV 字段的指针(因此也是虚拟地址)。此处需要，因为驱动程序在某些情况下会写入该字段，此外还会将其发送到加速器。

Pointer to (and therefore, the virtual address of) the additionalAuthData field above. Needed here because the driver in some cases writes to this field, in addition to sending it to the accelerator.
指向上面的 additionalAuthData 字段的指针(因此也是其虚拟地址)。此处需要，因为驱动程序在某些情况下会写入该字段，此外还会将其发送到加速器。

Opaque data that will be returned to the client in the function completion callback.
将在函数完成回调中返回给客户端的不透明数据。

This opaque data is not used by the implementation of the API, but is simply returned as part of the asynchronous response. It may be used to store information that might be useful when processing the response later.
API 的实现不使用这种不透明的数据，而只是作为异步响应的一部分返回。它可用于存储稍后处理响应时可能有用的信息。

9.6 Typedef Documentation

9.7 Typedef 文档

typedef void* CpaCySymDpSessionCtx

Cryptographic component symmetric session context handle for the data plane API.
数据平面 API 的加密组件对称会话上下文句柄。

Handle to a cryptographic data plane session context. The memory for this handle is allocated by the client. The size of the memory that the client needs to allocate is determined by a call to the **cpaCySymDpSessionCtxGetSize** or **cpaCySymDpSessionCtxGetDynamicSize** functions. The session context memory is initialized with a call to the **cpaCySymInitSession** function. This memory MUST not be freed until a call to **cpaCySymDpRemoveSession** has completed successfully.
加密数据层会话上下文的句柄。此句柄的内存由客户端分配。客户端需要分配的内存大小是通过调用 cpaCySymDpSessionCtxGetSize cpaCySymDpSessionCtxGetDynamicSize cpaCySymInitSession cpaCySymDpRemoveSession

typedef struct _CpaCySymDpOpData CpaCySymDpOpData

Operation Data for cryptographic data plane API.
加密数据平面 API 的操作数据。

This structure contains data relating to a request to perform symmetric cryptographic processing on one or more data buffers.
该结构包含与在一个或多个数据缓冲器上执行对称密码处理的请求相关的数据。

9.6 Typedef Documentation

The physical memory to which this structure points needs to be at least 8-byte aligned.

All reserved fields SHOULD NOT be written or read by the calling code.

这个结构指向的物理内存至少需要 8 字节对齐。调用代码不应写入或读取所有保留字

段。

See also:
另请参见:

cpaCySymDpEnqueueOp, **cpaCySymDpEnqueueOpBatch**
cpaCySymDpEnqueueOp, cpaCySymDpEnqueueOpBatch

Definition of callback function for cryptographic data plane API.
typedef void(* **CpaCySymDpCbFunc**)(**CpaCySymDpOpData** *pOpData, **CpaStatus** status, **CpaBoo**
加密数据平面 API 回调函数的定义。
typedef void(* CpaCySymDpCbFunc)(CpaCySymDpOpData *pOpData, CpaStatus status, CpaBoolean

This is the callback function prototype. The callback function is registered by the application using the **cpaCySymDpRegCbFunc** function call, and called back on completion of asycnhronous requests made via calls to **cpaCySymDpEnqueueOp** or **cpaCySymDpEnqueueOpBatch**.
这是回调函数原型。回调函数由应用程序使用 cpaCySymDpRegCbFunc cpaCySymDpEnqueueOp cpaCySymDpEnqueueOpBatch

Context:
背景:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.
这个回调函数可以在不允许休眠发生的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

No
不

Parameters:
参数:

[in] *pOpData* Pointer to the CpaCySymDpOpData object which was supplied as part of the original request.

[in]指向作为原始请求的一部分提供的 CpaCySymDpOpData 对象的 pOpData 指针。

[in] *status* Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.

[in] status 操作的状态。有效值为 CPA_STATUS_SUCCESS、CPA_STATUS_FAIL 和 CPA_STATUS_UNSUPPORTED。

[in] *verifyResult* This parameter is valid when the verifyDigest option is set in the CpaCySymSessionSetupData structure. A value of CPA_TRUE indicates that the compare succeeded. A value of CPA_FALSE indicates that the compare failed.

[in] verifyResult 当在 CpaCySymSessionSetupData 结构中设置了 verifyDigest 选项时，此参数有效。CPA_TRUE 值表示比较成功。CPA_FALSE 值表示比较失败。

Returns:

退货：

None
没有人

Precondition:

前提条件：

Component has been initialized. Callback has been registered with **cpaCySymDpRegCbFunc**.
组件已初始化。回拨已向注册 cpaCySymDpRegCbFunc

Postcondition:

后置条件：

None
没有人

Note:

注意：

None
没有人

See also:

另请参见：

cpaCySymDpRegCbFunc
cpaCySymDpRegCbFunc

9.8 Function Documentation

9.9 功能文档

CpaStatus cpaCySymDpRegCbFunc (const **CpaInstanceHandle**
CpaStatus cpaCySymDpRegCbFunc (const CpaInstanceHandle *instanceHandle*

Registration of the operation completion callback function.
操作完成回调函数的注册。

This function allows a completion callback function to be registered. The registered callback function is invoked on completion of asycnhronous requests made via calls to **cpaCySymDpEnqueueOp** or **cpaCySymDpEnqueueOpBatch**.
这个函数允许注册一个完成回调函数。注册的回调函数在通过调用 cpaCySymDpEnqueueOp cpaCySymDpEnqueueOpBatch

If a callback function was previously registered, it is overwritten.
如果之前注册了回调函数，它将被覆盖。

Context:
背景:

This is a synchronous function and it cannot sleep. It can be executed in a context that does not permit sleeping.
这是一个同步功能，它不能休眠。它可以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

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

No
不

Parameters:
参数:

[in] *instanceHandle* Instance on which the callback function is to be registered.
[in]要在其上注册回调函数的 instanceHandle 实例。

[in] *pSymNewCb* Callback function for this instance.
[in]这个执行个体的 pSymNewCb 回呼函式。

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_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。
CPA_STATUS_UNSUPPORTED Function is not supported.
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:

None
没有人

See also:

另请参见:

CpaCySymDpCbFunc
CpaCySymDpCbFunc

cpaCySymDpSessionCtxGetSize	(const const * *)	instanceHandle, pSessionSetupData, pSessionCtxSizeInBytes
cpaCySymDpSessionCtxGetSize	(const const * *)	instanceHandle, pSessionSetupData, pSessionCtxSizeInBytes

Gets the size required to store a session context for the data plane API.
获取存储数据平面 API 的会话上下文所需的大小。

This function is used by the client to determine the size of the memory it must allocate in order to store the session context. This MUST be called before the client allocates the memory for the session context and before the client calls the **cpaCySymDpInitSession** function.
客户端使用此函数来确定它必须分配的内存大小，以便存储会话上下文。在客户端为会话上下文分配内存之前，以及在客户端调用 cpaCySymDpInitSession

For a given implementation of this API, it is safe to assume that **cpaCySymDpSessionCtxGetSize()** will always return the same size and that the size will not be different for different setup data parameters.
对于该 API 的给定实现，可以安全地假设 cpaCySymDpSessionCtxGetSize()
However, it should be noted that the size may change: (1) between different implementations of the API (e.g. between software and hardware implementations or between different hardware implementations) (2) between different releases of the same API implementation.
然而，应当注意，大小可以改变:(1)在 API 的不同实现之间(例如，在软件和硬件实现之间或者在不同硬件实现之间)(2)在相同 API 实现的不同发布之间。

The size returned by this function is the smallest size needed to support all possible combinations of setup data parameters. Some setup data parameter combinations may fit within a smaller session context size. The alternate **cpaCySymDpSessionCtxGetDynamicSize()** function will return the smallest size needed to fit the provided setup data parameters.
此函数返回的大小是支持设置数据参数的所有可能组合所需的最小大小。一些设置数据参数组合可能适合较小的会话上下文大小。替补队员 cpaCySymDpSessionCtxGetDynamicSize()

Context:
背景:

This is a synchronous function that cannot sleep. It can be executed in a context that does not permit sleeping.
这是一个不能睡眠的同步功能。它可以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:

No
不

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] *pSessionSetupData* Pointer to session setup data which contains parameters which
[in]指向会话设置数据的 pSessionSetupData 指针，该数据包含
are static for a given cryptographic session such as operation type, mechanisms, and keys for cipher and/or hash operations.
对于给定加密会话是静态的，例如加密和/或散列操作的操作类型、机制和密钥。
[out] *pSessionCtxSizeInBytes* The amount of memory in bytes required to hold the Session
[out]psessiontxsizeinbytes 保存会话所需的内存量(以字节为单位)
Context.
语境。

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_UNSUPPORTED Function is not supported.

不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:
前提条件:

The component has been initialized.
组件已初始化。

Postcondition:
后置条件:

None
没有人

Note:

注意:

This is a synchronous function and has no completion callback associated with it.
这是一个同步函数，没有与之关联的完成回调。

See also:

另请参见:

CpaCySymSessionSetupData cpaCySymDpSessionCtxGetDynamicSize() cpaCySymDpInitSession()
CpaCySymSessionSetupData cpaCySymDpSessionCtxGetDynamicSize() cpaCySymDpInitSession()

CpaStatus

CpaStatus cpaCySymDpSessionCtxGetDynamicSize (const **CpaInstanceHandle** *instanceHandle*,
cpacysymdpsessiontxgetdynamicsize(const CpaInstanceHandle const *pSessionSetupData*,
常数
CpaCySymSessionSetupData *
CpaCySymSessionSetupData *

instanceHandle, *pSessionSetupData*,

Cpa32U * *pSessionCtxSizeInBytes*

Cpa32U * pSessionCtxSizeInBytes

)

)

Gets the minimum size required to store a session context for the data plane API.
获取存储数据平面 API 的会话上下文所需的最小大小。

This function is used by the client to determine the smallest size of the memory it must allocate in order to store the session context. This MUST be called before the client allocates the memory for the session context and before the client calls the **cpaCySymDpInitSession** function.
客户端使用此函数来确定它必须分配的最小内存大小，以便存储会话上下文。在客户端为会话上下文分配内存之前，以及在客户端调用 cpaCySymDpInitSession

This function is an alternate to cpaCySymDpSessionGetSize(). **cpaCySymDpSessionCtxGetSize()** will return a fixed size which is the minimum memory size needed to support all possible setup data parameter combinations. **cpaCySymDpSessionCtxGetDynamicSize()** will return the minimum memory size needed to support the specific session setup data parmeters provided. This size may be different for different setup data parameters.
此函数是 cpaCySymDpSessionGetSize()的替代函数。cpaCySymDpSessionCtxGetSize()
cpaCySymDpSessionCtxGetDynamicSize()

Context:
背景:

This is a synchronous function that cannot sleep. It can be executed in a context that does not permit sleeping.
这是一个不能睡眠的同步功能。它可以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:

No
不

Reentrant:
可重入:

No
不

Thread-safe:

线程安全:

Yes

是

Parameters:

[in]	*instanceHandle*	Instance handle.
[in]	*pSessionSetupData*	Pointer to session setup data which contains parameters which are static for a given cryptographic session such as operation

参数:

[在]	*instanceHandle*	实例句柄。
[在]	*pSessionSetupData*	指向会话设置数据的指针，该数据包含给定加密会话的静态参数，如操作

type, mechanisms, and keys for cipher and/or hash operations.

密码和/或哈希运算的类型、机制和密钥。

[out] *pSessionCtxSizeInBytes* The amount of memory in bytes required to hold the Session Context.

[out]psessiontxsizeinbytes 保存会话所需的内存量(以字节为单位)语境。

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:

The component has been initialized.

组件已初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

This is a synchronous function and has no completion callback associated with it.

这是一个同步函数，没有与之关联的完成回调。

See also:

另请参见:

CpaCySymSessionSetupData cpaCySymDpSessionCtxGetSize() cpaCySymDpInitSession()

CpaCySymSessionSetupData cpaCySymDpSessionCtxGetSize() cpaCySymDpInitSession()

```
cpaCySymDpInitSession (                                          instanceHandle
cpaCySymDpInitSession (                                          instanceHandle,
                                const *  pSessionSetupData,
                                sessionCtx
                              )
```

Initialize a session for the symmetric cryptographic data plane API.

初始化对称加密数据平面 API 的会话。

This function is used by the client to initialize an asynchronous session context for symmetric cryptographic data plane operations. The returned session context is the handle to the session and needs to be passed when requesting cryptographic operations to be performed.
客户端使用此函数为对称加密数据平面操作初始化异步会话上下文。返回的会话上下文是会话的句柄，需要在请求执行加密操作时传递。

Only sessions created using this function may be used when invoking functions on this API

The session can be removed using **cpaCySymDpRemoveSession**.

当调用此 API 上的函数时，只能使用使用此函数创建的会话。可以使用

cpaCySymDpRemoveSession

Context:
背景:

This is a synchronous function and it cannot sleep. It can be executed in a context that does not permit sleeping.
这是一个同步功能，它不能休眠。它可以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:

No
不

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Parameters:

参数:

[in] *instanceHandle* Instance to which the requests will be submitted.
[in]请求将提交到的 instanceHandle 实例。

[in] *pSessionSetupData* Pointer to session setup data which contains parameters that are
[in]指向会话设置数据的 pSessionSetupData 指针，该数据包含以下参数
static for a given cryptographic session such as operation type, algorithm, and keys for cipher and/or hash operations.
给定加密会话的静态信息，例如操作类型、算法以及加密和/或哈希操作的密钥。

[out] *sessionCtx* Pointer to the memory allocated by the client to store the session
[out] sessionCtx 指向客户端为存储会话而分配的内存的指针
context. This memory must be physically contiguous, and its length (in bytes) must be at least as big as specified by a call to **cpaCySymDpSessionCtxGetSize**. This memory will be initialized with this function. This value needs to be passed to subsequent processing calls.
语境。该内存在物理上必须是连续的，并且其长度(以字节为单位)必须至少与调用 cpaCySymDpSessionCtxGetSize

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.
组件已初始化。

Postcondition:

后置条件:

None
没有人

Note:

注意:

This is a synchronous function and has no completion callback associated with it.
这是一个同步函数，没有与之关联的完成回调。

See also:

另请参见：

cpaCySymDpSessionCtxGetSize, **cpaCySymDpRemoveSession**

cpaCySymDpSessionCtxGetSize, cpaCySymDpRemoveSession

CpaStatus cpaCySymDpRemoveSession (const **CpaInstanceHandle**

CpaStatus cpaCySymDpRemoveSession(const CpaInstanceHandle *instanceHandle*,

CpaCySymDpSessionCtx **sessionCtx*

Remove (delete) a symmetric cryptographic session for the data plane API.

移除(删除)数据平面 API 的对称加密会话。

This function will remove a previously initialized session context and the installed callback handler function. Removal will fail if outstanding calls still exist for the initialized session handle. The client needs to retry the remove function at a later time. The memory for the session context MUST not be freed until this call has completed successfully.

此函数将删除先前初始化的会话上下文和已安装的回调处理函数。如果初始化的会话句柄仍存在未完成的调用，移除将会失败。客户端需要稍后重试删除功能。在此调用成功完成之前，不得释放会话上下文的内存。

Context:

背景：

This is a synchronous function that cannot sleep. It can be executed in a context that does not permit sleeping.

这是一个不能睡眠的同步功能。它可以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：

Reentrant:

可重入:

No
不

Thread-safe:
线程安全:

No
不

Parameters:
参数:

[in] *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。
[in,out] *sessionCtx* Session context to be removed.
[in, out] sessionCtx 要移除的会话上下文。

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.
组件已初始化。

Postcondition:
后置条件:

None
没有人

Note:

注意:

Note that this is a synchronous function and has no completion callback associated with it.
请注意，这是一个同步函数，没有与之关联的完成回调。

See also:

另请参见:

CpaCySymDpSessionCtx, **cpaCySymDpInitSession()**
CpaCySymDpSessionCtx, cpaCySymDpInitSession()

Enqueue a single symmetric cryptographic request.
将单个对称加密请求排队。

```
CpaStatus cpaCySymDpEnqueueOp(CpaCySymDpOpData *pOpData,
    const CpaBoolean performOpNow);
```

This function enqueues a single request to perform a cipher, hash or combined (cipher and hash) operation. Optionally, the request is also submitted to the cryptographic engine to be performed.
此函数将单个请求排入队列，以执行加密、哈希或组合(加密和哈希)操作。可选地，该请求也被提交给密码引擎来执行。

See note about performance trade-offs on the **Symmetric cryptographic Data Plane API** API.
请参阅上关于性能权衡的注释 Symmetric cryptographic Data Plane API

The function is asynchronous; control is returned to the user once the request has been submitted. On completion of the request, the application may poll for responses, which will cause a callback function (registered via **cpaCySymDpRegCbFunc**) to be invoked. Callbacks within a session are guaranteed to be in the same order in which they were submitted.
该函数是异步的；一旦提交了请求，控制权就返回给用户。请求完成后，应用程序可能会轮询响应，这将导致回调函数(通过 cpaCySymDpRegCbFunc

The following restrictions apply to the pOpData parameter:
下列限制适用于 pOpData 参数:

- The memory MUST be aligned on an 8-byte boundary.
- 内存必须在 8 字节边界上对齐。
- The structure MUST reside in physically contiguous memory.
- 该结构必须驻留在物理上连续的内存中。
- The reserved fields of the structure SHOULD NOT be written or read by the calling code.
- 调用代码不应写入或读取该结构的保留字段。

Context:
背景:

This function will not sleep, and hence can be executed in a context that does not permit sleeping.
这个函数不会休眠，因此可以在不允许休眠的上下文中执行。

Side-Effects:

副作用：
None
没有人

Blocking:
阻止：
No
不

Reentrant:
可重入：
No
不

Thread-safe:
线程安全：
No
不

Parameters:
参数：

[in] *pOpData* Pointer to a structure containing the request parameters. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback, which was registered on the instance via **cpaCySymDpRegCbFunc**. See the above Description for restrictions that apply to this parameter.

[in]指向包含请求参数的结构的 pOpData 指针。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，回调是通过在实例上注册的 cpaCySymDpRegCbFunc

[in] *performOpNow* Flag to specify whether the operation should be performed immediately (CPA_TRUE), or simply enqueued to be performed later (CPA_FALSE). In the latter case, the request is submitted to be performed either by calling this function again with this flag set to CPA_TRUE, or by invoking the function **cpaCySymDpPerformOpNow**.

[in] performOpNow 标志，用于指定是应该立即执行该操作(CPA_TRUE)，还是只是将其排入队列以便稍后执行(CPA_FALSE)。在后一种情况下，请求被提交执行，要么通过将该标志设置为 CPA_TRUE 再次调用该函数，要么通过调用该函数 cpaCySymDpPerformOpNow

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_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。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 session identified by pOpData->sessionCtx was setup using **cpaCySymDpInitSession**. The instance identified by pOpData->instanceHandle has had a callback function registered via **cpaCySymDpRegCbFunc**.

由 pOpData->sessionCtx 标识的会话是使用 cpaCySymDpInitSessioncpaCySymDpRegCbFunc

Postcondition:

后置条件:

None

没有人

Note:

注意:

A callback of type **CpaCySymDpCbFunc** is generated in response to this function call. Any errors generated during processing are reported as part of the callback status code.
A callb类型的回调 CpaCySymDpCbFunc

See also:

另请参见:

cpaCySymDpInitSession, **cpaCySymDpPerformOpNow**
cpaCySymDpInitSession, cpaCySymDpPerformOpNow

```
CpaStatus
cpaCySymDpEnqueueOpBatch (  const Cpa32U numberRequests,
    const CpaCySymDpOpData * pOpData[], const CpaBoolean performOpNow
)
```

Enqueue multiple requests to the symmetric cryptographic data plane API.

将多个请求排入对称加密数据平面 API 的队列。

This function enqueues multiple requests to perform cipher, hash or combined (cipher and hash) operations.
此函数将多个请求排队以执行加密、哈希或组合(加密和哈希)操作。

See note about performance trade-offs on the **Symmetric cryptographic Data Plane API** API.
请参阅上关于性能权衡的注释 Symmetric cryptographic Data Plane API

The function is asynchronous; control is returned to the user once the request has been submitted. On completion of the request, the application may poll for responses, which will cause a callback function (registered via **cpaCySymDpRegCbFunc**) to be invoked. Separate callbacks will be invoked for each request. Callbacks within a session are guaranteed to be in the same order in which they were submitted.
该函数是异步的；一旦提交了请求，控制权就返回给用户。请求完成后，应用程序可能会轮询响应，这将导致回调函数(通过 cpaCySymDpRegCbFunc

The following restrictions apply to each element of the pOpData array:
下列限制适用于 pOpData 数组的每个元素:

- The memory MUST be aligned on an 8-byte boundary.
- 内存必须在 8 字节边界上对齐。
- The structure MUST reside in physically contiguous memory.
- 该结构必须驻留在物理上连续的内存中。
- The reserved fields of the structure SHOULD NOT be written or read by the calling code.
- 调用代码不应写入或读取该结构的保留字段。

Context:
背景:

This function will not sleep, and hence can be executed in a context that does not permit sleeping.
这个函数不会休眠，因此可以在不允许休眠的上下文中执行。

Assumptions:
假设:

Client MUST allocate the request parameters to 8 byte alignment. Reserved elements of the CpaCySymDpOpData structure MUST be 0. The CpaCySymDpOpData structure MUST reside in physically contiguous memory.
客户端必须将请求参数分配给 8 字节对齐。CpaCySymDpOpData 结构的保留元素必须为 0。CpaCySymDpOpData 结构必须驻留在物理上连续的内存中。

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

No

不

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

No
不

Parameters:
参数:

[in] *numberRequests* The number of requests in the array of CpaCySymDpOpData structures.
[in]number requests CpaCySymDpOpData 结构数组中的请求数。

[in] *pOpData* An array of pointers to CpaCySymDpOpData structures. Each of the CpaCySymDpOpData structure contains the request parameters for that request. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback, which was registered on the instance via **cpaCySymDpRegCbFunc**. See the above Description for restrictions that apply to this parameter.

[in] pOpData 指向 CpaCySymDpOpData 结构的指标阵列。每个 CpaCySymDpOpData 结构都包含该请求的请求参数。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，回调是通过在实例上注册的 cpaCySymDpRegCbFunc

[in] *performOpNow* Flag to specify whether the operation should be performed immediately (CPA_TRUE), or simply enqueued to be performed later (CPA_FALSE). In the latter case, the request is submitted to be performed either by calling this function again with this flag set to CPA_TRUE, or by invoking the function **cpaCySymDpPerformOpNow**.

[in] performOpNow 标志，用于指定是应该立即执行该操作(CPA_TRUE)，还是只是将其排入队列以便稍后执行(CPA_FALSE)。在后一种情况下，请求被提交执行，要么通过将该标志设置为 CPA_TRUE 再次调用该函数，要么通过调用该函数 cpaCySymDpPerformOpNow

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_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。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 session identified by pOpData[i]->sessionCtx was setup using **cpaCySymDpInitSession**. The instance identified by pOpData->instanceHandle[i] has had a callback function registered via **cpaCySymDpRegCbFunc**.
由 pOpData[i]->sessionCtx 标识的会话是使用 cpaCySymDpInitSessioncpaCySymDpRegCbFunc

Postcondition:
后置条件:

None
没有人

Note:

注意:

ple callbacks of type **CpaCySymDpCbFunc** are generated in response to this function call (one per request). Any errors generated during processing are reported as part of the callback status code.

Multi 类型的多个回调 CpaCySymDpCbFunc

See also:

另请参见:

cpaCySymDpInitSession, **cpaCySymDpEnqueueOp**

cpaCySymDpInitSession, cpaCySymDpEnqueueOp

CpaStatus cpaCySymDpPerformOpNow (**CpaInstanceHandle** *instanceHandle*)

Submit any previously enqueued requests to be performed now on the symmetric cryptographic data plane API.

CpaStatuCpaInstanceHandl

s cpaCySymDpPerformOpNow(

If any requests/operations were enqueued via calls to **cpaCySymDpEnqueueOp** and/or **cpaCySymDpEnqueueOpBatch**, but with the flag performOpNow set to **CPA_FALSE**, then these operations will now be submitted to the accelerator to be performed.

如果有任何请求/操作通过调用 cpaCySymDpEnqueueOp cpaCySymDpEnqueueOpBatchCPA_FALSE

See note about performance trade-offs on the **Symmetric cryptographic Data Plane API** API.

请参阅上关于性能权衡的注释 Symmetric cryptographic Data Plane API

Context:

背景:

Will not sleep. It can be executed in a context that does not permit sleeping.

不会睡觉。它可以在不允许休眠的上下文中执行。

Side-Effects:

副作用:

None

没有人

Blocking:

阻止:

No

不

Reentrant:

可重入:

No

不

Thread-safe:

线程安全:

No
不

Parameters:
参数:

[in] *instanceHandle* Instance to which the requests will be submitted.
[in]请求将提交到的 instanceHandle 实例。

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

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. A cryptographic session has been previously setup using the
组件已初始化。先前已经使用设置了加密会话
cpaCySymDpInitSession function call.
cpaCySymDpInitSession 函数调用。

Postcondition:
后置条件:

None
没有人

See also:
另请参见:

cpaCySymDpEnqueueOp, **cpaCySymDpEnqueueOpBatch**
cpaCySymDpEnqueueOp, cpaCySymDpEnqueueOpBatch

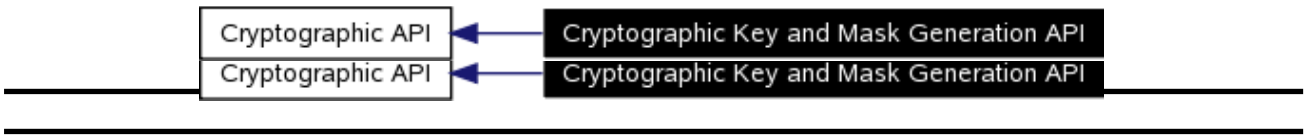
10 Cryptographic Key and Mask Generation API

11 加密密钥和掩码生成 API

[Cryptographic API]

[Cryptographic API]

Collaboration diagram for Cryptographic Key and Mask Generation API:
加密密钥和掩码生成 API 的协作图:



11.1 Detailed Description

11.2 详细描述

File: cpa_cy_key.h

文件:cpa_cy_key.h

These functions specify the API for key and mask generation operations.
这些函数指定了键和掩码生成操作的 API。

11.3 Data Structures

11.4 数据结构

- struct _**CpaCyKeyGenSslOpData**
- 结构体_CpaCyKeyGenSslOpData
- struct _**CpaCyKeyGenHKDFExpandLabel**
- 结构体_CpaCyKeyGenHKDFExpandLabel
- struct _**CpaCyKeyGenHKDFOpData**
- 结构体_CpaCyKeyGenHKDFOpData
- struct _**CpaCyKeyGenTlsOpData**
- 结构体_CpaCyKeyGenTlsOpData
- struct _**CpaCyKeyGenMgfOpData**
- 结构体_CpaCyKeyGenMgfOpData
- struct _**CpaCyKeyGenMgfOpDataExt**
- 结构体_CpaCyKeyGenMgfOpDataExt
- struct _**CpaCyKeyGenStats**
- 结构体_CpaCyKeyGenStats
- struct _**CpaCyKeyGenStats64**

- 结构体_CpaCyKeyGenStats64

11.5 Defines
11.6 界定

- #define **CPA_CY_KEY_GEN_SSL_TLS_RANDOM_LEN_IN_BYTES**

- #定义 CPA_CY_KEY_GEN_SSL_TLS_RANDOM_LEN_IN_BYTES
- #define **CPA_CY_HKDF_SUBLABEL_KEY**
- #定义 CPA_CY_HKDF_SUBLABEL_KEY

11.7 Typedefs
11.8 类型定义

- typedef enum _**CpaCyKeySslOp CpaCyKeySslOp**

- typedef 枚举_CpaCyKeySslOp CpaCyKeySslOp
- typedef _**CpaCyKeyGenSslOpData CpaCyKeyGenSslOpData**
- 数据类型说明_CpaCyKeyGenSslOpData CpaCyKeyGenSslOpData
- typedef enum _**CpaCyKeyTlsOp CpaCyKeyTlsOp**
- typedef 枚举_CpaCyKeyTlsOp CpaCyKeyTlsOp
- typedef enum _**CpaCyKeyHKDFOp CpaCyKeyHKDFOp**
- typedef 枚举_CpaCyKeyHKDFOp CpaCyKeyHKDFOp
- typedef enum _**CpaCyKeyHKDFCipherSuite CpaCyKeyHKDFCipherSuite**
- typedef 枚举_CpaCyKeyHKDFCipherSuite CpaCyKeyHKDFCipherSuite
- typedef _**CpaCyKeyGenHKDFExpandLabel CpaCyKeyGenHKDFExpandLabel**
- 数据类型说明_CpaCyKeyGenHKDFExpandLabel CpaCyKeyGenHKDFExpandLabel
- typedef _**CpaCyKeyGenHKDFOpData CpaCyKeyGenHKDFOpData**
- 数据类型说明_CpaCyKeyGenHKDFOpData CpaCyKeyGenHKDFOpData
- typedef _**CpaCyKeyGenTlsOpData CpaCyKeyGenTlsOpData**
- 数据类型说明_CpaCyKeyGenTlsOpData CpaCyKeyGenTlsOpData
- typedef _**CpaCyKeyGenMgfOpData CpaCyKeyGenMgfOpData**
- 数据类型说明_CpaCyKeyGenMgfOpData CpaCyKeyGenMgfOpData
- typedef _**CpaCyKeyGenMgfOpDataExt CpaCyKeyGenMgfOpDataExt**
- 数据类型说明_CpaCyKeyGenMgfOpDataExt CpaCyKeyGenMgfOpDataExt
- typedef _**CpaCyKeyGenStats CPA_DEPRECATED**
- 数据类型说明_CpaCyKeyGenStats CPA_DEPRECATED
- typedef _**CpaCyKeyGenStats64 CpaCyKeyGenStats64**
- 数据类型说明_CpaCyKeyGenStats64 CpaCyKeyGenStats64

11.9 Enumerations
11.10 列举

- enum _**CpaCyKeySslOp** {
  CPA_CY_KEY_SSL_OP_MASTER_SECRET_DERIVE,
  CPA_CY_KEY_SSL_OP_KEY_MATERIAL_DERIVE,

- 列举型别_CpaCyKeySslOp
  CPA_CY_KEY_SSL_OP_MASTER_SECRET_DERIVECPA_CY_KEY_S
  SL_OP_KEY_MATERIAL_DERIVE

10.5 Enumerations

10.6 列举

CPA_CY_KEY_SSL_OP_USER_DEFINED

CPA_CY_KEY_SSL_OP_USER_DEFINED
}
}

- enum _**CpaCyKeyTlsOp** {
  CPA_CY_KEY_TLS_OP_MASTER_SECRET_DERIVE,
  CPA_CY_KEY_TLS_OP_KEY_MATERIAL_DERIVE,
  CPA_CY_KEY_TLS_OP_CLIENT_FINISHED_DERIVE,
  CPA_CY_KEY_TLS_OP_SERVER_FINISHED_DERIVE,
  CPA_CY_KEY_TLS_OP_USER_DEFINED
- 列举型别_CpaCyKeyTlsOp
  CPA_CY_KEY_TLS_OP_MASTER_SECRET_DERIVECPA_CY_KEY_TL
  S_OP_KEY_MATERIAL_DERIVECPA_CY_KEY_TLS_OP_CLIENT_FI
  NISHED_DERIVECPA_CY_KEY_TLS_OP_SERVER_FINISHED_DERI
  VECPA_CY_KEY_TLS_OP_USER_DEFINED
  }
  }
- enum _**CpaCyKeyHKDFOp** {
  CPA_CY_HKDF_KEY_EXTRACT,
  CPA_CY_HKDF_KEY_EXPAND,
  CPA_CY_HKDF_KEY_EXTRACT_EXPAND,
  CPA_CY_HKDF_KEY_EXPAND_LABEL,
  CPA_CY_HKDF_KEY_EXTRACT_EXPAND_LABEL
- 列举型别_CpaCyKeyHKDFOp
  CPA_CY_HKDF_KEY_EXTRACTCPA_CY_HKDF_KEY_EXPANDCP
  A_CY_HKDF_KEY_EXTRACT_EXPANDCPA_CY_HKDF_KEY_EXP
  AND_LABELCPA_CY_HKDF_KEY_EXTRACT_EXPAND_LABEL
  }
  }
- enum _**CpaCyKeyHKDFCipherSuite** {
  CPA_CY_HKDF_TLS_AES_128_GCM_SHA256,
  CPA_CY_HKDF_TLS_AES_256_GCM_SHA384,
  CPA_CY_HKDF_TLS_CHACHA20_POLY1305_SHA256,
  CPA_CY_HKDF_TLS_AES_128_CCM_SHA256,
  CPA_CY_HKDF_TLS_AES_128_CCM_8_SHA256
- **列举型别**_CpaCyKeyHKDFCipherSuite
  }
  }

10.7 Functions

10.8 功能

- **CpaStatus cpaCyKeyGenSsl** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyKeyGenSsl（常量 CpaInstanceHandle
  CpaCyGenFlatBufCbFunc pKeyGenCb, void *pCallbackTag, const **CpaCyKeyGenSslOpData**
  CpaCyGenFlatBufCbFunc pKeyGenCb, void *pCallbackTag, constCpaCyKeyGenSslOpData
  *pKeyGenSslOpData, **CpaFlatBuffer** *pGeneratedKeyBuffer)
  * pKeyGenSslOpData, CpaFlatBuffer
- **CpaStatus cpaCyKeyGenTls** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyKeyGenTls（常量 CpaInstanceHandle
  CpaCyGenFlatBufCbFunc pKeyGenCb, void *pCallbackTag, const **CpaCyKeyGenTlsOpData**
  CpaCyGenFlatBufCbFunc pKeyGenCb, void *pCallbackTag, constCpaCyKeyGenTlsOpData
  *pKeyGenTlsOpData, **CpaFlatBuffer** *pGeneratedKeyBuffer)

*pKeyGenTlsOpData, CpaFlatBuffer

- **CpaStatus cpaCyKeyGenTls2** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyKeyGenTls2 (常量 CpaInstanceHandle
  CpaCyGenFlatBufCbFunc pKeyGenCb, void *pCallbackTag, const **CpaCyKeyGenTlsOpData**
  CpaCyGenFlatBufCbFunc pKeyGenCb, void *pCallbackTag, constCpaCyKeyGenTlsOpData
  *pKeyGenTlsOpData, **CpaCySymHashAlgorithm** hashAlgorithm, **CpaFlatBuffer**
  *pKeyGenTlsOpData, CpaCySymHashAlgorithm CpaFlatBuffer
  *pGeneratedKeyBuffer)
  *pGeneratedKeyBuffer)
- **CpaStatus cpaCyKeyGenTls3** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyKeyGenTls3 (常量 CpaInstanceHandle
  CpaCyGenFlatBufCbFunc pKeyGenCb, void *pCallbackTag, const **CpaCyKeyGenHKDFOpData**
  CpaCyGenFlatBufCbFunc pKeyGenCb, void *pCallbackTag, constCpaCyKeyGenHKDFOpData
  *pKeyGenTlsOpData, **CpaCyKeyHKDFCipherSuite** cipherSuite, **CpaFlatBuffer**
  *pKeyGenTlsOpData, CpaCyKeyHKDFCipherSuite CpaFlatBuffer
  *pGeneratedKeyBuffer)
  *pGeneratedKeyBuffer)
- **CpaStatus cpaCyKeyGenMgf** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyKeyGenMgf (常量 CpaInstanceHandle
  CpaCyGenFlatBufCbFunc pKeyGenCb, void *pCallbackTag, const **CpaCyKeyGenMgfOpData**
  CpaCyGenFlatBufCbFunc pKeyGenCb, void *pCallbackTag, constCpaCyKeyGenMgfOpData
  *pKeyGenMgfOpData, **CpaFlatBuffer** *pGeneratedMaskBuffer)
  *pKeyGenMgfOpData, CpaFlatBuffer
- **CpaStatus cpaCyKeyGenMgfExt** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyKeyGenMgfExt (常量 CpaInstanceHandle
  CpaCyGenFlatBufCbFunc pKeyGenCb, void *pCallbackTag, const **CpaCyKeyGenMgfOpDataExt**
  CpaCyGenFlatBufCbFunc pKeyGenCb, void *pCallbackTag, constCpaCyKeyGenMgfOpDataExt
  *pKeyGenMgfOpDataExt, **CpaFlatBuffer** *pGeneratedMaskBuffer)
  *pKeyGenMgfOpDataExt, CpaFlatBuffer
- **CpaStatus CPA_DEPRECATED cpaCyKeyGenQueryStats** (const **CpaInstanceHandle**
- CpaStatus CPA_DEPRECATED cpaCyKeyGenQueryStats (常量 CpaInstanceHandle
  instanceHandle, struct **_CpaCyKeyGenStats** *pKeyGenStats)
  instanceHandle, 结构_CpaCyKeyGenStats
- **CpaStatus cpaCyKeyGenQueryStats64** (const **CpaInstanceHandle** instanceHandle,
- CpaStatus cpaCyKeyGenQueryStats64 (常量 CpaInstanceHandle
  CpaCyKeyGenStats64 *pKeyGenStats)
  CpaCyKeyGenStats64 *pKeyGenStats)

10.9 Data Structure Documentation

10.10 数据结构文档

10.7 Data Structure Documentation

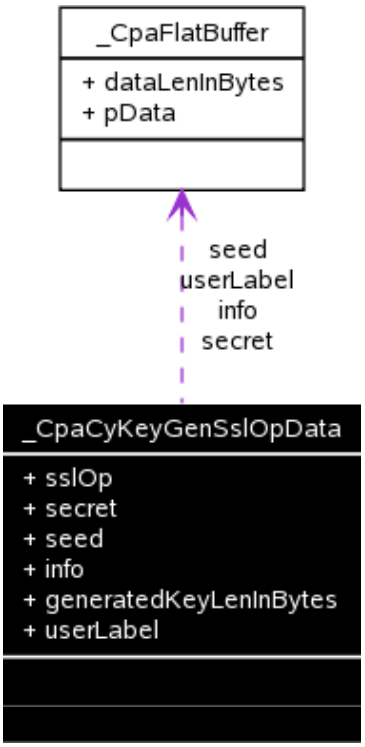
10.8 数据结构文档

10.8.1 _CpaCyKeyGenSslOpData Struct Reference

10.8.2 _ CpaCyKeyGenSslOpData 结构引用

Collaboration diagram for _CpaCyKeyGenSslOpData:

_ CpaCyKeyGenSslOpData 的协作图:



10.8.2.1 Detailed Description
10.8.2.2 详细描述

SSL data for key generation functions
密钥生成函数的 SSL 数据

This structure contains data for use in key generation operations for SSL. For specific SSL key generation operations, the structure fields MUST be set as follows:
此结构包含用于 SSL 密钥生成操作的数据。对于特定的 SSL 密钥生成操作，结构字段必须设置如下:

SSL Master-Secret Derivation:
SSL 主密钥派生:

sslOp = CPA_CY_KEY_SSL_OP_MASTER_SECRET_DERIVE

ss lop = CPA _ CY _ KEY _ SSL _ OP _ MASTER _ SECRET _ DERIVE
secret = pre-master secret key
secret =预主密钥
seed = client_random + server_random
userLabel = NULL
seed =客户端随机+服务器随机用户标签
=空

SSL Key-Material Derivation:
SSL 密钥-材料派生:

sslOp = CPA_CY_KEY_SSL_OP_KEY_MATERIAL_DERIVE
ss lop = CPA _ CY _ KEY _ SSL _ OP _ KEY _ MATERIAL _ DERIVE
secret = master secret key
secret =主密钥
seed = server_random + client_random
userLabel = NULL
seed =服务器随机+客户端随机用户标签
=空

Note that the client/server random order is reversed from that used for master-secret derivation.

请注意，客户端/服务器随机顺序与用于主密钥推导的顺序相反。

Note:
注意:

Each of the client and server random numbers need to be of length CPA_CY_KEY_GEN_SSL_TLS_RANDOM_LEN_IN_BYTES.
每个客户端和服务器随机数的长度都必须是CPA _ CY _ KEY _ GEN _ SSL _ TLS _ RANDOM _ LEN _ IN _ BYTES。

10.7.1 _CpaCyKeyGenSslOpData Struct Reference

10.7.2_ CpaCyKeyGenSslOpData 结构引用

In each of the above descriptions, + indicates concatenation.

在上面的每个描述中，+表示连接。

The label used is predetermined by the SSL operation in line with the SSL 3.0 specification, and can be overridden by using a user defined operation CPA_CY_KEY_SSL_OP_USER_DEFINED and associated userLabel.
使用的标签由符合 SSL 3.0 规范的 SSL 操作预先确定，并且可以通过使用用户定义的操作 CPA_CY_KEY_SSL_OP_USER_DEFINED 和关联的 userLabel 来覆盖。

10.8.2.3 Data Fields
10.8.2.4 数据字段

- **CpaCyKeySslOp sslOp**
- CpaCyKeySslOp sslOp
- **CpaFlatBuffer secret**
- CpaFlatBuffer secret
- **CpaFlatBuffer seed**
- CpaFlatBuffer seed
- **CpaFlatBuffer info**
- CpaFlatBuffer info
- **Cpa32U generatedKeyLenInBytes**
- Cpa32U generatedKeyLenInBytes
- **CpaFlatBuffer userLabel**
- CpaFlatBuffer userLabel

10.8.2.5 Field Documentation
10.8.2.6 现场文件

Indicate the SSL operation to be performed
指示要执行的 SSL 操作

Flat buffer containing a pointer to either the master or pre-master secret key. The length field indicates the length of the secret key in bytes. Implementation-specific limits may apply to this length.
包含指向主密钥或预主密钥的指针的平面缓冲区。长度字段以字节表示密钥的长度。特定于实现的限制可能适用于该长度。

Flat buffer containing a pointer to the seed data. Implementation-specific limits may apply to this length.
包含种子数据指针的平面缓冲区。特定于实现的限制可能适用于该长度。

Flat buffer containing a pointer to the info data. Implementation-specific limits may apply to this length.
包含指向信息数据的指针的平面缓冲区。特定于实现的限制可能适用于该长度。

The requested length of the generated key in bytes. Implementation-specific limits may apply to this length.
生成的密钥的请求长度(以字节为单位)。特定于实现的限制可能适用于该长度。

Optional flat buffer containing a pointer to a user defined label. The length field indicates the length of the label in bytes. To use this field, the sslOp must be CPA_CY_KEY_SSL_OP_USER_DEFINED, or otherwise it is ignored and can be set to NULL. Implementation-specific limits may apply to this length.
可选的平面缓冲区，包含一个指向用户定义标签的指针。长度字段以字节表示标签的长度。要使用该字段，sslOp 必须是 CPA_CY_KEY_SSL_OP_USER_DEFINED，否则它将被忽略并可设置为 NULL。特定于实现的限制可能适用于该长度。

10.7.3 _CpaCyKeyGenHKDFExpandLabel Struct Reference

10.7.4 _CpaCyKeyGenHKDFExpandLabel 结构引用

10.7.4.1 Detailed Description

File: cpa_cy_key.h

10.7.4.2 详细描述文

件:cpa_cy_key.h

TLS data for key generation functions
密钥生成函数的 TLS 数据

This structure contains data for describing label for the HKDF Extract Label function
此结构包含用于描述 HKDF 提取标签功能的标签的数据

Extract Label Function
提取标签功能

labelLen = length of the label field
contextLen = length of the context field
labelLen =标签字段长度 contextLen =上下文字段长度

10.7.2 _CpaCyKeyGenHKDFExpandLabel Struct Reference

10.7.3_CpaCyKeyGenHKDFExpandLabel 结构引用

sublabelFlag = Mask of sub labels required for this label.
label = label as defined in RFC8446

sublabelFlag =该标签所需的子标签的掩码。标签= RFC 8446 中定义的标签
context = context as defined in RFC8446
上下文= RFC 8446 中定义的上下文

10.7.4.3 Data Fields
10.7.4.4 数据字段

- **Cpa8U label** [CPA_CY_HKDF_KEY_MAX_LABEL_SZ]
- Cpa8U label [CPA_CY_HKDF_KEY_MAX_LABEL_SZ]
- **Cpa8U labelLen**
- Cpa8U labelLen
- **Cpa8U sublabelFlag**
- Cpa8U sublabelFlag

10.7.4.5 Field Documentation
10.7.4.6 现场文件

Cpa8U _CpaCyKeyGenHKDFExpandLabel::label[CPA_CY_HKDF_KEY_MAX_LABEL_SZ]

HKDFLabel field as defined in RFC8446 sec 7.1.
RFC8446 第 7.1 节中定义的 HKDFLabel 字段。

Cpa8U _CpaCyKeyGenHKDFExpandLabel::labelLen

The length, in bytes of the label
标签的长度，以字节为单位

Cpa8U _CpaCyKeyGenHKDFExpandLabel::sublabelFlag

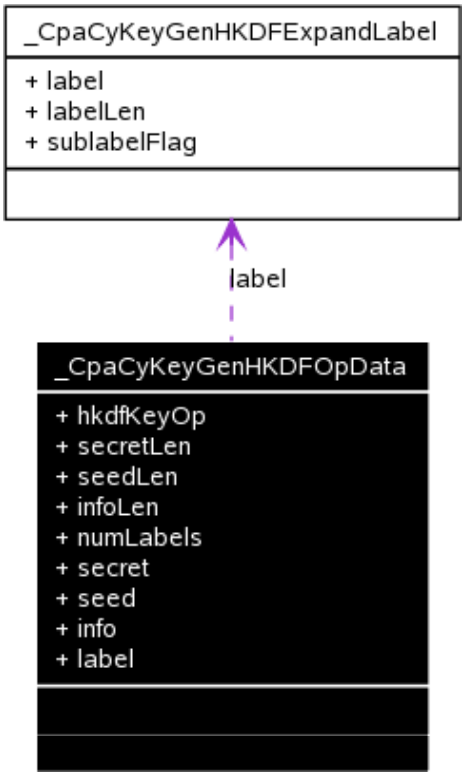
mask of sublabels to be generated. This flag is composed of zero or more of:
CPA_CY_HKDF_SUBLABEL_KEY CPA_CY_HKDF_SUBLABEL_IV
CPA_CY_HKDF_SUBLABEL_RESUMPTION CPA_CY_HKDF_SUBLABEL_FINISHED
要生成的子标签的掩码。这个标志由零个或多个组成:CPA _ CY _ HKDF _子标签_ KEY CPA _ CY _ HKDF _子标签_ IV CPA _ CY _ HKDF _子标签_ RESUMPTION CPA _ CY _ HKDF _子标签_FINISHED

10.7.4 _CpaCyKeyGenHKDFOpData Struct Reference

10.7.5 _CpaCyKeyGenHKDFOpData 结构引用

Collaboration diagram for _CpaCyKeyGenHKDFOpData:

_CpaCyKeyGenHKDFOpData 的协作图:


_CpaCyKeyGenHKDFExpandLabel
+ label
+ labelLen
+ sublabelFlag
label
_CpaCyKeyGenHKDFOpData
+ hkdfKeyOp
+ secretLen
+ seedLen
+ infoLen
+ numLabels
+ secret
+ seed
+ info
+ label

10.7.3 _CpaCyKeyGenHKDFOpData Struct Reference

10.7.4_CpaCyKeyGenHKDFOpData 结构引用

10.7.4.1 Detailed Description

10.7.4.2 详细描述

TLS data for key generation functions
密钥生成函数的 TLS 数据

This structure contains data for all HKDF operations:
此结构包含所有 HKDF 操作的数据:
HKDF Extract
HKDF Expand
HKDF 提取物
HKDF 扩展
HKDF Expand Label
HKDF Extract and Expand
HKDF 扩展标签 HKDF 提取和
扩展
HKDF Extract and Expand Label
HKDF 提取和扩展标签

HKDF Map Structure Elements
HKDF 地图结构元素

secret - IKM value for extract operations or PRK for expand or expand operations.
seed - contains the salt for extract operations
secret - IKM 值用于提取操作，PRK 值用于扩展或扩充操作。种子-包含用于提取操作的盐
info - contains the info data for extract operations
labels - See notes above
信息-包含提取操作标签的信息数据-参见上述注释

10.7.4.3 Data Fields
10.7.4.4 数据字段

- **CpaCyKeyHKDFOp hkdfKeyOp**
- CpaCyKeyHKDFOp hkdfKeyOp
- **Cpa8U secretLen**
- Cpa8U secretLen
- **Cpa16U seedLen**
- Cpa16U seedLen
- **Cpa16U infoLen**
- Cpa16U infoLen
- **Cpa16U numLabels**
- Cpa16U numLabels
- **Cpa8U secret** [CPA_CY_HKDF_KEY_MAX_SECRET_SZ]
- Cpa8U secret [CPA _ CY _ HKDF _ KEY _ MAX _ SECRET _ SZ]
- **Cpa8U seed** [CPA_CY_HKDF_KEY_MAX_HMAC_SZ]
- Cpa8U seed [CPA _ CY _ HKDF _ KEY _ MAX _ HMAC _深圳]
- **Cpa8U info** [CPA_CY_HKDF_KEY_MAX_INFO_SZ]

- Cpa8U info [CPA_CY_HKDF_KEY_MAX_INFO_SZ]
- **CpaCyKeyGenHKDFExpandLabel label** [CPA_CY_HKDF_KEY_MAX_LABEL_COUNT]
- CpaCyKeyGenHKDFExpandLabel label [CPA _ CY _ HKDF _ KEY _ MAX _ LABEL _ COUNT]

10.7.4.5 Field Documentation
10.7.4.6 现场文件

Keying operation to be performed.
要执行的键控操作。

Length of secret field
秘密字段的长度

Length of seed field
种子场长度

Length of info field
信息字段的长度

Number of filled CpaCyKeyGenHKDFExpandLabel elements
已填充的 CpaCyKeyGenHKDFExpandLabel 元素的数量

Input Key Material or PRK
输入关键材料或 PRK

Input salt
输入盐

Cpa8U _CpaCyKeyGenHKDFOpData::info[CPA_CY_HKDF_KEY_MAX_INFO_SZ]

info field

Cpa8_CpaCyKeyGenHKDFOpData::info

array of Expand **CpaCyKeyGenHKDFExpandLabel**

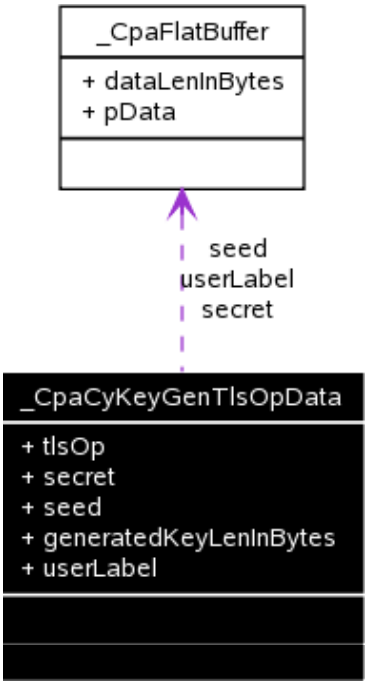
扩展标签结构阵列 CpaCyKeyGenHKDFExpandLabel[CPA_CY_HKDF_KEY_MAX_LABEL_COUNT]

10.7.5 _CpaCyKeyGenTlsOpData Struct Reference

10.7.6 _CpaCyKeyGenTlsOpData 结构引用

Collaboration diagram for _CpaCyKeyGenTlsOpData:

_CpaCyKeyGenTlsOpData 的协作图:



10.7.6.1 Detailed Description
10.7.6.2 详细描述

TLS data for key generation functions
密钥生成函数的 TLS 数据

This structure contains data for use in key generation operations for TLS. For specific TLS key generation operations, the structure fields MUST be set as follows:
此结构包含用于 TLS 密钥生成操作的数据。对于特定的 TLS 密钥生成操作，结构字段必须设置如下:

TLS Master-Secret Derivation:
TLS 主密钥派生：

tlsOp = CPA_CY_KEY_TLS_OP_MASTER_SECRET_DERIVE
tlsOp = CPA _ CY _ KEY _ TLS _ OP _ MASTER _ SECRET _ DERIVE
secret = pre-master secret key
secret =预主密钥
seed = client_random + server_random
userLabel = NULL
seed =客户端随机+服务器随机用户标签
=空

TLS Key-Material Derivation:
TLS 密钥-材料推导：

tlsOp = CPA_CY_KEY_TLS_OP_KEY_MATERIAL_DERIVE
tlsOp = CPA _ CY _ KEY _ TLS _ OP _ KEY _ MATERIAL _ DERIVE
secret = master secret key
secret =主密钥
seed = server_random + client_random
userLabel = NULL
seed =服务器随机+客户端随机用户标签
=空

Note that the client/server random order is reversed from that used for Master-Secret Derivation.
请注意，客户端/服务器随机顺序与用于主密钥推导的顺序相反。

TLS Client finished/Server finished tag Derivation:
TLS 客户端完成/服务器完成标记派生：

tlsOp = CPA_CY_KEY_TLS_OP_CLIENT_FINISHED_DERIVE (client)
or CPA_CY_KEY_TLS_OP_SERVER_FINISHED_DERIVE (server)
secret = master secret key
tlsOp = CPA _ CY _ KEY _ TLS _ OP _ CLIENT _ FINISHED _ DERIVE(客户端)或CPA _ CY _ KEY _ TLS _ OP _ SERVER _ FINISHED _ DERIVE(服务器)secret =主密钥
seed = MD5(handshake_messages) + SHA-1(handshake_messages)
userLabel = NULL
种子= MD5(握手消息)+ SHA-1(握手消息)用户标签=空

Note:
注意：

Each of the client and server random seeds need to be of length CPA_CY_KEY_GEN_SSL_TLS_RANDOM_LEN_IN_BYTES.
每个客户端和服务器随机种子的长度都必须是 CPA _ CY _ KEY _ GEN _ SSL _ TLS _ RANDOM _ LEN _ IN _ BYTES。

In each of the above descriptions, + indicates concatenation.
在上面的每个描述中，+表示连接。

The label used is predetermined by the TLS operation in line with the TLS specifications, and can be overridden by using a user defined operation CPA_CY_KEY_TLS_OP_USER_DEFINED and associated userLabel.
使用的标签由 TLS 操作根据 TLS 规范预先确定，并且可以通过使用用户定义的操作 CPA_CY_KEY_TLS_OP_USER_DEFINED 和关联的 userLabel 来覆盖。

10.7.6.3 Data Fields

10.7.6.4 数据字段

- **CpaCyKeyTlsOp tlsOp**
- CpaCyKeyTlsOp tlsOp
- **CpaFlatBuffer secret**
- CpaFlatBuffer secret
- **CpaFlatBuffer seed**
- CpaFlatBuffer seed
- **Cpa32U generatedKeyLenInBytes**
- Cpa32U generatedKeyLenInBytes
- **CpaFlatBuffer userLabel**
- CpaFlatBuffer userLabel

10.7.6.5 Field Documentation

10.7.6.6 现场文件

TLS operation to be performed
要执行的 TLS 操作

Flat buffer containing a pointer to either the master or pre-master secret key. The length field indicates the length of the secret in bytes.
包含指向主密钥或预主密钥的指针的平面缓冲区。长度字段以字节表示秘密的长度。

Flat buffer containing a pointer to the seed data. Implementation-specific limits may apply to this length.
包含种子数据指针的平面缓冲区。特定于实现的限制可能适用于该长度。

The requested length of the generated key in bytes. Implementation-specific limits may apply to this length.
生成的密钥的请求长度(以字节为单位)。特定于实现的限制可能适用于该长度。

Optional flat buffer containing a pointer to a user defined label. The length field indicates the length of the label in bytes. To use this field, the tlsOp must be CPA_CY_KEY_TLS_OP_USER_DEFINED.

Implementation-specific limits may apply to this length.

可选的平面缓冲区，包含一个指向用户定义标签的指针。长度字段以字节表示标签的长度。要使用该字段，tlsOp 必须是 CPA_CY_KEY_TLS_OP_USER_DEFINED。特定于实现的限制可能适用于该长度。

10.7.7 _CpaCyKeyGenMgfOpData Struct Reference

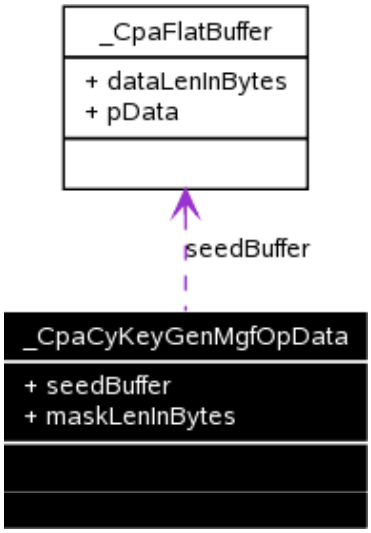
10.7.8_CpaCyKeyGenMgfOpData 结构引用

10.7.5 _CpaCyKeyGenMgfOpData Struct Reference

10.7.6 _CpaCyKeyGenMgfOpData 结构引用

Collaboration diagram for _CpaCyKeyGenMgfOpData:

_CpaCyKeyGenMgfOpData 的协作图:



10.7.6.1 Detailed Description
10.7.6.2 详细描述

Key Generation Mask Generation Function (MGF) Data
密钥生成掩码生成函数(MGF)数据

This structure contains data relating to Mask Generation Function key generation operations.
该结构包含与掩码生成功能键生成操作相关的数据。

Note:
注意:

The default hash algorithm used by the MGF is SHA-1. If a different hash algorithm is preferred, then see the extended version of this structure, **CpaCyKeyGenMgfOpDataExt**.
MGF 使用的默认哈希算法是 SHA-1。如果首选不同的哈希算法，那么请参见此结构的扩展版本，CpaCyKeyGenMgfOpDataExt

See also:
另请参见:

cpaCyKeyGenMgf
cpaCyKeyGenMgf

10.7.6.3 Data Fields
10.7.6.4 数据字段

- **CpaFlatBuffer seedBuffer**
- CpaFlatBuffer seedBuffer
- **Cpa32U maskLenInBytes**
- Cpa32U maskLenInBytes

10.7.6.5 Field Documentation
10.7.6.6 现场文件

CpaFlatBuffer _CpaCyKeyGenMgfOpData::seedBuffer

Caller MUST allocate a buffer and populate with the input seed data. For optimal performance the start of the seed SHOULD be allocated on an 8-byte boundary. The length field represents the seed length in CpaFlatBuffe_CpaCyKeyGenMgfOpData::seedBuffe bytes. Implementation-specific limits may apply to this length.
字节。特定于实现的限制可能适用于该长度。

Cpa32U _CpaCyKeyGenMgfOpData::maskLenInBytes

The requested length of the generated mask in bytes. Implementation-specific limits may apply to this length.
Cpa32_CpaCyKeyGenMgfOpData::maskLenInByte

10.7.7 _CpaCyKeyGenMgfOpDataExt Struct Reference

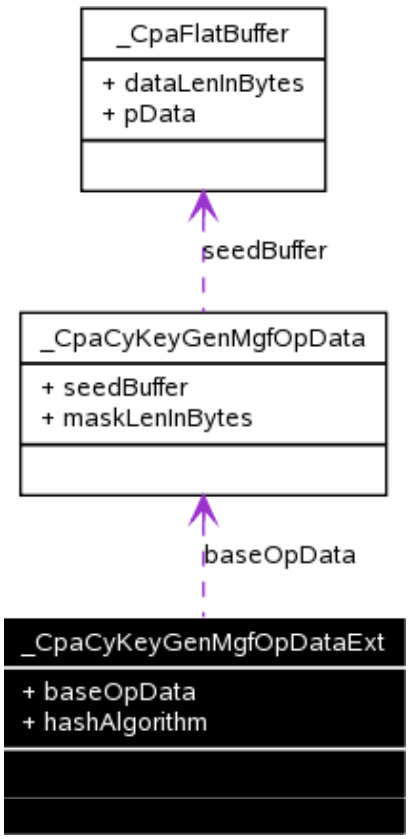
10.7.8_CpaCyKeyGenMgfOpDataExt 结构引用

_CpaCyKeyGenMgfOpDataExt Struct Reference

_CpaCyKeyGenMgfOpDataExt 结构引用

Collaboration diagram for _CpaCyKeyGenMgfOpDataExt:

_CpaCyKeyGenMgfOpDataExt 的协作图:



10.7.7.1 Detailed Description
10.7.7.2 详细描述

Extension to the original Key Generation Mask Generation Function (MGF) Data
对原始密钥生成掩码生成函数(MGF)数据的扩展

This structure is an extension to the original MGF data structure. The extension allows the hash function to be specified.
这种结构是对原始 MGF 数据结构的扩展。该扩展允许指定哈希函数。

Note:
注意:

This structure is separate from the base **CpaCyKeyGenMgfOpData** structure in order to retain

backwards compatibility with the original version of the API.
该结构与基座分离 CpaCyKeyGenMgfOpData

See also:
另请参见:

cpaCyKeyGenMgfExt
cpaCyKeyGenMgfExt

10.7.7.3 Data Fields
10.7.7.4 数据字段

- **CpaCyKeyGenMgfOpData baseOpData**
- CpaCyKeyGenMgfOpData baseOpData
- **CpaCySymHashAlgorithm hashAlgorithm**
- CpaCySymHashAlgorithm hashAlgorithm

10.7.7.5 Field Documentation
10.7.7.6 现场文件

CpaCyKeyGenMgfOpData _CpaCyKeyGenMgfOpDataExt::baseOpData
CpaCyKeyGenMgfOpDat_CpaCyKeyGenMgfOpDataExt::baseOpDat

10.7.8 _CpaCyKeyGenStats Struct Reference

10.7.9_CpaCyKeyGenStats 结构引用

"Base" operational data for MGF generation

MGF 一代的“基本”操作数据

Specifies the hash algorithm to be used by the Mask Generation Function

指定掩码生成函数要使用的哈希算法

10.7.7 _CpaCyKeyGenStats Struct Reference

10.7.8 _CpaCyKeyGenStats 结构引用

10.7.8.1 Detailed Description

Key Generation Statistics.

Deprecated:

10.7.8.2 密钥生成统计信息的

详细描述。Deprecated:

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by **CpaCyKeyGenStats64**.

从 Crypto API 的 1.3 版开始，这种结构已被取代，由 CpaCyKeyGenStats64

This structure contains statistics on the key and mask generation operations. Statistics are set to zero when the component is initialized, and are collected per instance.

该结构包含键和掩码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。

10.7.8.3 Data Fields

10.7.8.4 数据字段

- **Cpa32U numSslKeyGenRequests**
- Cpa32U numSslKeyGenRequests
- **Cpa32U numSslKeyGenRequestErrors**
- Cpa32U numSslKeyGenRequestErrors
- **Cpa32U numSslKeyGenCompleted**
- Cpa32U numSslKeyGenCompleted
- **Cpa32U numSslKeyGenCompletedErrors**
- Cpa32U numSslKeyGenCompletedErrors
- **Cpa32U numTlsKeyGenRequests**
- Cpa32U numTlsKeyGenRequests
- **Cpa32U numTlsKeyGenRequestErrors**
- Cpa32U numTlsKeyGenRequestErrors
- **Cpa32U numTlsKeyGenCompleted**
- Cpa32U numTlsKeyGenCompleted
- **Cpa32U numTlsKeyGenCompletedErrors**
- Cpa32U numTlsKeyGenCompletedErrors
- **Cpa32U numMgfKeyGenRequests**
- Cpa32U numMgfKeyGenRequests

- **Cpa32U numMgfKeyGenRequestErrors**
- Cpa32U numMgfKeyGenRequestErrors
- **Cpa32U numMgfKeyGenCompleted**
- Cpa32U numMgfKeyGenCompleted
- **Cpa32U numMgfKeyGenCompletedErrors**
- Cpa32U numMgfKeyGenCompletedErrors

10.7.8.5 Field Documentation
10.7.8.6 现场文件

Total number of successful SSL key generation requests.
成功的 SSL 密钥生成请求的总数。

Total number of SSL key generation requests that had an error and could not be processed.
出现错误且无法处理的 SSL 密钥生成请求的总数。

Total number of SSL key generation operations that completed successfully.
成功完成的 SSL 密钥生成操作的总数。

Total number of SSL key generation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 SSL 密钥生成操作的总数。

Total number of successful TLS key generation requests.
成功的 TLS 密钥生成请求的总数。

Total number of TLS key generation requests that had an error and could not be processed.

出现错误且无法处理的 TLS 密钥生成请求的总数。

Total number of TLS key generation operations that completed successfully.
成功完成的 TLS 密钥生成操作的总数。

Total number of TLS key generation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 TLS 密钥生成操作的总数。

Total number of successful MGF key generation requests (including "extended" MGF requests).
成功的 MGF 密钥生成请求的总数(包括“扩展的”MGF 请求)。

Total number of MGF key generation requests that had an error and could not be processed.
出现错误且无法处理的 MGF 密钥生成请求的总数。

Total number of MGF key generation operations that completed successfully.
成功完成的 MGF 密钥生成操作的总数。

Total number of MGF key generation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 MGF 密钥生成操作的总数。

10.7.8 _CpaCyKeyGenStats64 Struct Reference

10.7.9 _CpaCyKeyGenStats64 结构引用

10.7.9.1 Detailed Description
10.7.9.2 详细描述

Key Generation Statistics (64-bit version).
密钥生成统计信息(64 位版本)。

This structure contains the 64-bit version of the statistics on the key and mask generation operations. Statistics are set to zero when the component is initialized, and are collected per instance.
此结构包含 64 位版本的键和掩码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。

10.7.9.3 Data Fields
10.7.9.4 数据字段

- **Cpa64U numSslKeyGenRequests**
- Cpa64U numSslKeyGenRequests
- **Cpa64U numSslKeyGenRequestErrors**

- Cpa64U numSslKeyGenRequestErrors
- **Cpa64U numSslKeyGenCompleted**
- Cpa64U numSslKeyGenCompleted
- **Cpa64U numSslKeyGenCompletedErrors**
- Cpa64U numSslKeyGenCompletedErrors
- **Cpa64U numTlsKeyGenRequests**
- Cpa64U numTlsKeyGenRequests
- **Cpa64U numTlsKeyGenRequestErrors**
- Cpa64U numTlsKeyGenRequestErrors
- **Cpa64U numTlsKeyGenCompleted**
- Cpa64U numTlsKeyGenCompleted
- **Cpa64U numTlsKeyGenCompletedErrors**
- Cpa64U numTlsKeyGenCompletedErrors
- **Cpa64U numMgfKeyGenRequests**
- Cpa64U numMgfKeyGenRequests
- **Cpa64U numMgfKeyGenRequestErrors**
- Cpa64U numMgfKeyGenRequestErrors
- **Cpa64U numMgfKeyGenCompleted**
- Cpa64U numMgfKeyGenCompleted
- **Cpa64U numMgfKeyGenCompletedErrors**
- Cpa64U numMgfKeyGenCompletedErrors

10.7.9.5 Field Documentation
10.7.9.6 现场文件

Cpa64U CpaCyKeyGenStats64::numSslKeyGenRequests

Total number of successful SSL key generation requests.
成功的 SSL 密钥生成请求的总数。

Cpa64U CpaCyKeyGenStats64::numSslKeyGenRequestErrors

Total number of SSL key generation requests that had an error and could not be processed.
出现错误且无法处理的 SSL 密钥生成请求的总数。

Total number of SSL key generation operations that completed successfully.
成功完成的 SSL 密钥生成操作的总数。

Total number of SSL key generation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 SSL 密钥生成操作的总数。

Total number of successful TLS key generation requests.
成功的 TLS 密钥生成请求的总数。

Total number of TLS key generation requests that had an error and could not be processed.
出现错误且无法处理的 TLS 密钥生成请求的总数。

Total number of TLS key generation operations that completed successfully.
成功完成的 TLS 密钥生成操作的总数。

Total number of TLS key generation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 TLS 密钥生成操作的总数。

Total number of successful MGF key generation requests (including "extended" MGF requests).
成功的 MGF 密钥生成请求的总数(包括“扩展的”MGF 请求)。

Total number of MGF key generation requests that had an error and could not be processed.
出现错误且无法处理的 MGF 密钥生成请求的总数。

Total number of MGF key generation operations that completed successfully.
成功完成的 MGF 密钥生成操作的总数。

Total number of MGF key generation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 MGF 密钥生成操作的总数。

10.8 Define Documentation

10.9 定义文档

SSL or TLS key generation random number length.
SSL 或 TLS 密钥生成随机数长度。

Defines the permitted SSL or TLS random number length in bytes that may be used with the functions
定义函数可能使用的允许的 SSL 或 TLS 随机数长度(以字节为单位)
cpaCyKeyGenSsl and **cpaCyKeyGenTls**. This is the length of the client or server random number values.
cpaCyKeyGenSsl 和 cpaCyKeyGenTls

File: cpa_cy_key.h

文件:cpa_cy_key.h

TLS Operation Types
TLS 操作类型

Bitwise constants for HKDF sublabels
HKDF 子标签的按位常数

These definitions provide bit settings for sublabels for HKDF-ExpandLabel operations.
这些定义为 HKDF 扩展标签操作的子标签提供了位设置。

key sublabel to generate "key" keying material

生成“密钥”密钥材料的密钥子标签

iv sublabel to generate "iv" keying material

生成“iv”密钥材料的 iv 子标签
resumption sublabel to generate "resumption" keying material
用于生成“恢复”密钥材料的恢复子标签
finished sublabel to generate "finished" keying material Bit for creation of key material for 'key' sublabel
成品子标签生成“成品”密钥材料位，用于创建“密钥”子标签的密钥材料

10.10 Typedef Documentation

10.11 Typedef 文档

SSL Operation Types
SSL 操作类型

Enumeration of the different SSL operations that can be specified in the struct **CpaCyKeyGenSslOpData**. It identifies the label.
结构中可以指定的不同 SSL 操作的枚举 CpaCyKeyGenSslOpData

SSL data for key generation functions
密钥生成函数的 SSL 数据

This structure contains data for use in key generation operations for SSL. For specific SSL key generation operations, the structure fields MUST be set as follows:
此结构包含用于 SSL 密钥生成操作的数据。对于特定的 SSL 密钥生成操作，结构字段必须设置如下:

SSL Master-Secret Derivation:
SSL 主密钥派生:

sslOp = CPA_CY_KEY_SSL_OP_MASTER_SECRET_DERIVE
ss lop = CPA _ CY _ KEY _ SSL _ OP _ MASTER _ SECRET _ DERIVE
secret = pre-master secret key
secret =预主密钥
seed = client_random + server_random
userLabel = NULL
seed =客户端随机+服务器随机用户标签
=空

SSL Key-Material Derivation:
SSL 密钥-材料派生:

sslOp = CPA_CY_KEY_SSL_OP_KEY_MATERIAL_DERIVE
ss lop = CPA _ CY _ KEY _ SSL _ OP _ KEY _ MATERIAL _ DERIVE
secret = master secret key
secret =主密钥
seed = server_random + client_random
userLabel = NULL

10.0 Typedef

seed =服务器随机+客户端随机用户标签
=空

Note that the client/server random order is reversed from that used for master-secret derivation.

请注意，客户端/服务器随机顺序与用于主密钥推导的顺序相反。

Note:
注意:

Each of the client and server random numbers need to be of length CPA_CY_KEY_GEN_SSL_TLS_RANDOM_LEN_IN_BYTES.
每个客户端和服务器随机数的长度都必须是 CPA _ CY _ KEY _ GEN _ SSL _ TLS _ RANDOM _ LEN _ IN _ BYTES。

In each of the above descriptions, + indicates concatenation.
在上面的每个描述中，+表示连接。

The label used is predetermined by the SSL operation in line with the SSL 3.0 specification, and can be overridden by using a user defined operation CPA_CY_KEY_SSL_OP_USER_DEFINED and associated userLabel.
使用的标签由符合 SSL 3.0 规范的 SSL 操作预先确定，并且可以通过使用用户定义的操作 CPA_CY_KEY_SSL_OP_USER_DEFINED 和关联的 userLabel 来覆盖。

typedef enum **CpaCyKeyTlsOp CpaCyKeyTlsOp**

TLS Operation Types
TLS 操作类型

Enumeration of the different TLS operations that can be specified in the CpaCyKeyGenTlsOpData. It identifies the label.
可以在 CpaCyKeyGenTlsOpData 中指定的不同 TLS 操作的枚举。它识别标签。

The functions **cpaCyKeyGenTls** and **cpaCyKeyGenTls2** accelerate the TLS PRF, which is defined as part of RFC2246 (TLS v1.0), RFC4346 (TLS v1.1), and RFC5246 (TLS v1.2). One of the inputs to each of these
这些功能 cpaCyKeyGenTls cpaCyKeyGenTls2

functions is a label. This enumerated type defines values that correspond to some of the required labels. However, for some of the operations/labels required by these RFCs, no values are specified.

函数是一个标签。该枚举类型定义了对应于某些必需标签的值。但是，对于这些 RFC 要求的一些操作/标签，没有指定值。

In such cases, a user-defined value must be provided. The client should use the enum value **CPA_CY_KEY_TLS_OP_USER_DEFINED**, and pass the label using the userLabel field of the **CpaCyKeyGenTlsOpData** data structure.
在这种情况下，必须提供用户定义的值。客户端应该使用枚举值
CPA_CY_KEY_TLS_OP_USER_DEFINEDCpaCyKeyGenTlsOpData

typedef enum _**CpaCyKeyHKDFOp CpaCyKeyHKDFOp**

File: cpa_cy_key.h

typedef 枚举_CpaCyKeyHKDFOCpaCyKeyHKDFO

TLS Operation Types
TLS 操作类型

Enumeration of the different TLS operations that can be specified in the CpaCyKeyGenHKDFOpData.
可以在 CpaCyKeyGenHKDFOpData 中指定的不同 TLS 操作的枚举。

The function **cpaCyKeyGenTls3** accelerates the TLS HKDF, which is defined as part of RFC5869 (HKDF) and RFC8446 (TLS v1.3).
该功能 cpaCyKeyGenTls3

This enumerated type defines the support HKDF operations for extraction and expansion of keying material.
此枚举类型定义了提取和扩展密钥材料的支持 HKDF 操作。

typedef enum _**CpaCyKeyHKDFCipherSuite CpaCyKeyHKDFCipherSuite**

File: cpa_cy_key.h

typedef 枚举_CpaCyKeyHKDFCipherSuitCpaCyKeyHKDFCipherSuit

TLS Operation Types
TLS 操作类型

Enumeration of the different cipher suites that may be used in a TLS v1.3 operation. This value is used to infer the sizes of the key and iv sublabel.
TLS v1.3 操作中可能使用的不同密码套件的枚举。该值用于推断键和 iv 子标签的大小。

The function **cpaCyKeyGenTls3** accelerates the TLS HKDF, which is defined as part of RFC5869 (HKDF) and RFC8446 (TLS v1.3).
该功能 cpaCyKeyGenTls3

This enumerated type defines the supported cipher suites in the TLS operation that require HKDF key operations.
此枚举类型定义了需要 HKDF 密钥操作的 TLS 操作中支持的密码套件。

typedef struct _**CpaCyKeyGenHKDFExpandLabel CpaCyKeyGenHKDFExpandLabel**

File: cpa_cy_key.h

typedef 结构_CpaCyKeyGenHKDFExpandLabeCpaCyKeyGenHKDFExpandLabe

TLS data for key generation functions
密钥生成函数的 TLS 数据

This structure contains data for describing label for the HKDF Extract Label function
此结构包含用于描述 HKDF 提取标签功能的标签的数据

Extract Label Function
提取标签功能

labelLen = length of the label field
contextLen = length of the context field
labelLen =标签字段长度 contextLen =
上下文字段长度
sublabelFlag = Mask of sub labels required for this label.
label = label as defined in RFC8446
sublabelFlag =该标签所需的子标签的掩码。标签= RFC
8446 中定义的标签
context = context as defined in RFC8446
上下文= RFC 8446 中定义的上下文

typedef struct _**CpaCyKeyGenHKDFOpData CpaCyKeyGenHKDFOpData**
typedef 结构_CpaCyKeyGenHKDFOpDatCpaCyKeyGenHKDFOpDat

TLS data for key generation functions
密钥生成函数的 TLS 数据

This structure contains data for all HKDF operations:

此结构包含所有 HKDF 操作的数据:
HKDF Extract
HKDF Expand
HKDF 提取物
HKDF 扩展
HKDF Expand Label
HKDF Extract and Expand
HKDF 扩展标签 HKDF 提取和
扩展
HKDF Extract and Expand Label
HKDF 提取和扩展标签

HKDF Map Structure Elements
HKDF 地图结构元素

secret - IKM value for extract operations or PRK for expand or expand operations.
secret - IKM 值用于提取操作，PRK 值用于扩展或扩充操作。
seed - contains the salt for extract operations
info - contains the info data for extract operations
labels - See notes above
seed -包含提取操作信息的 salt 包含提取操作标
签的信息数据-参见上面的注释

typedef struct _**CpaCyKeyGenTlsOpData CpaCyKeyGenTlsOpData**
typedef 结构_CpaCyKeyGenTlsOpDatCpaCyKeyGenTlsOpDat

TLS data for key generation functions
密钥生成函数的 TLS 数据

This structure contains data for use in key generation operations for TLS. For specific TLS key generation operations, the structure fields MUST be set as follows:
此结构包含用于 TLS 密钥生成操作的数据。对于特定的 TLS 密钥生成操作，结构字段必须设置如下:

TLS Master-Secret Derivation:
TLS 主密钥派生:

tlsOp = CPA_CY_KEY_TLS_OP_MASTER_SECRET_DERIVE
tlsOp = CPA _ CY _ KEY _ TLS _ OP _ MASTER _ SECRET _ DERIVE
secret = pre-master secret key
secret =预主密钥
seed = client_random + server_random
userLabel = NULL
seed =客户端随机+服务器随机用户标签
=空

TLS Key-Material Derivation:
TLS 密钥-材料推导:

tlsOp = CPA_CY_KEY_TLS_OP_KEY_MATERIAL_DERIVE
tlsOp = CPA _ CY _ KEY _ TLS _ OP _ KEY _ MATERIAL _ DERIVE
secret = master secret key
secret =主密钥
seed = server_random + client_random
userLabel = NULL
seed =服务器随机+客户端随机用户标签
=空

Note that the client/server random order is reversed from that used for Master-Secret Derivation.

请注意，客户端/服务器随机顺序与用于主密钥推导的顺序相反。

TLS Client finished/Server finished tag Derivation:
TLS 客户端完成/服务器完成标记派生：

tlsOp = CPA_CY_KEY_TLS_OP_CLIENT_FINISHED_DERIVE (client)
or CPA_CY_KEY_TLS_OP_SERVER_FINISHED_DERIVE (server)
secret = master secret key
tlsOp = CPA _ CY _ KEY _ TLS _ OP _ CLIENT _ FINISHED _
DERIVE(客户端)或CPA _ CY _ KEY _ TLS _ OP _ SERVER _ FINISHED
_ DERIVE(服务器)secret =主密钥
seed = MD5(handshake_messages) + SHA-1(handshake_messages)
userLabel = NULL
种子= MD5(握手消息)+ SHA-1(握手消息)用户标签=空

Note:
注意：

length CPA_CY_KEY_GEN_SSL_TLS_RANDOM_LEN_IN_BYTES.
每个客户端和服务器随机种子的长度都必须是 CPA _ CY _ KEY _ GEN _ SSL _ TLS _ RANDOM _ LEN _ IN _ BYTES。

Each of the client and server random seeds need to be of

In each of the above descriptions, + indicates concatenation.
在上面的每个描述中，+表示连接。

The label used is predetermined by the TLS operation in line with the TLS specifications, and can be overridden by using a user defined operation CPA_CY_KEY_TLS_OP_USER_DEFINED and associated userLabel.
使用的标签由 TLS 操作根据 TLS 规范预先确定，并且可以通过使用用户定义的操作 CPA_CY_KEY_TLS_OP_USER_DEFINED 和关联的 userLabel 来覆盖。

typedef struct **CpaCyKeyGenMgfOpData CpaCyKeyGenMgfOpData**

Key Generation Mask Generation Function (MGF) Data
密钥生成掩码生成函数(MGF)数据

This structure contains data relating to Mask Generation Function key generation operations.
该结构包含与掩码生成功能键生成操作相关的数据。

Note:
注意:

The default hash algorithm used by the MGF is SHA-1. If a different hash algorithm is preferred, then see the extended version of this structure, **CpaCyKeyGenMgfOpDataExt**.
MGF 使用的默认哈希算法是 SHA-1。如果首选不同的哈希算法，那么请参见此结构的扩展版本，CpaCyKeyGenMgfOpDataExt

See also:
另请参见:

cpaCyKeyGenMgf
cpaCyKeyGenMgf

typedef struct **CpaCyKeyGenMgfOpDataExt CpaCyKeyGenMgfOpDataExt**

Extension to the original Key Generation Mask Generation Function (MGF) Data
对原始密钥生成掩码生成函数(MGF)数据的扩展

This structure is an extension to the original MGF data structure. The extension allows the hash function to be specified.
这种结构是对原始 MGF 数据结构的扩展。该扩展允许指定哈希函数。

Note:
注意:

This structure is separate from the base **CpaCyKeyGenMgfOpData** structure in order to retain backwards compatibility with the original version of the API.
该结构与基座分离 CpaCyKeyGenMgfOpData

See also:
另请参见:

cpaCyKeyGenMgfExt
cpaCyKeyGenMgfExt

typedef struct **CpaCyKeyGenStats CPA_DEPRECATED**

Key Generation Statistics.
密钥生成统计信息。

Deprecated:
Deprecated:

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by
从 Crypto API 的 1.3 版开始，这种结构已被取代，由
CpaCyKeyGenStats64.
CpaCyKeyGenStats64。

This structure contains statistics on the key and mask generation operations. Statistics are set to zero when the component is initialized, and are collected per instance.
该结构包含键和掩码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。

Key Generation Statistics (64-bit version).
密钥生成统计信息(64 位版本)。

This structure contains the 64-bit version of the statistics on the key and mask generation operations. Statistics are set to zero when the component is initialized, and are collected per instance.
此结构包含 64 位版本的键和掩码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。

10.12 Enumeration Type Documentation

10.13 枚举类型文档

SSL Operation Types
SSL 操作类型

Enumeration of the different SSL operations that can be specified in the struct **CpaCyKeyGenSslOpData**. It identifies the label.
结构中可以指定的不同 SSL 操作的枚举 CpaCyKeyGenSslOpData

Enumerator:
枚举器:

CPA_CY_KEY_SSL_OP_MASTER_SECRET_DERIVE	Derive the master secret
CPA_CY_KEY_SSL_OP_KEY_MATERIAL_DERIVE	Derive the key material
CPA_CY_KEY_SSL_OP_USER_DEFINED	User Defined Operation for custom labels
CPA_CY_KEY_SSL_OP_MASTER_SECRET_DERIVE	*导出主密钥 CPA_CY_KEY_SSL_OP_KEY_MATERIAL_DERIVE* 导出密钥材料
CPA_CY_KEY_SSL_OP_USER_DEFINED	自定义标签的用户定义操作

enum _**CpaCyKeyTlsOp**

列举型别_CpaCyKeyTlsO

TLS Operation Types

TLS 操作类型

Enumeration of the different TLS operations that can be specified in the CpaCyKeyGenTlsOpData. It identifies the label.

可以在 CpaCyKeyGenTlsOpData 中指定的不同 TLS 操作的枚举。它识别标签。

The functions **cpaCyKeyGenTls** and **cpaCyKeyGenTls2** accelerate the TLS PRF, which is defined as part of RFC2246 (TLS v1.0), RFC4346 (TLS v1.1), and RFC5246 (TLS v1.2). One of the inputs to each of these functions is a label. This enumerated type defines values that correspond to some of the required labels.

这些功能 cpaCyKeyGenTls cpaCyKeyGenTls2

However, for some of the operations/labels required by these RFCs, no values are specified.

但是，对于这些 RFC 要求的一些操作/标签，没有指定值。

In such cases, a user-defined value must be provided. The client should use the enum value **CPA_CY_KEY_TLS_OP_USER_DEFINED**, and pass the label using the userLabel field of the **CpaCyKeyGenTlsOpData** data structure.

在这种情况下，必须提供用户定义的值。客户端应该使用枚举值

CPA_CY_KEY_TLS_OP_USER_DEFINEDCpaCyKeyGenTlsOpData

Enumerator:

枚举器:

CPA_CY_KEY_TLS_OP_MASTER_SECRET_DERIVE Derive the master secret using the TLS PRF. Corresponds to RFC2246/5246 section 8.1, operation "Computing the master secret", label "master secret".

CPA _ CY _ KEY _ TLS _ OP _ MASTER _ SECRET _ DERIVE 使用 TLS 导出主密钥

PRF 。 对 应 于 RFC2246/5246 第 8.1 节“计算主密钥”操作，标签为“主密钥”。

CPA_CY_KEY_TLS_OP_KEY_MATERIAL_DERIVE Derive the key material using the TLS PRF. Corresponds to RFC2246/5246 section 6.3, operation "Derive the key material", label "key expansion".

CPA _ CY _ KEY _ TLS _ OP _ KEY _ MATERIAL _ DERIVE 使用 TLS 导出密钥材料

PRF 。 对 应 于 RFC2246/5246 第 6.3 节，操作“导出密钥材料”，标签“密钥扩展”。

CPA_CY_KEY_TLS_OP_CLIENT_FINISHED_DERIVE Derive the client finished tag using the TLS PRF. Corresponds to RFC2246/5246 section 7.4.9, operation "Client finished", label "client finished".

CPA _ CY _ KEY _ TLS _ OP _ CLIENT _ FINISHED _ DERIVE 使用

TLS PRF。对应于 RFC2246/5246 第 7.4.9 节，操作“客户端完成”，标签“客户端完成”。

CPA_CY_KEY_TLS_OP_SERVER_FINISHED_DERIVE Derive the server finished tag using the TLS PRF. Corresponds to RFC2246/5246 section 7.4.9, operation

CPA _ CY _ KEY _ TLS _ OP _ SERVER _ FINISHED _ DERIVE 使用

"Server finished", label "server finished".
TLS PRF。对应于 RFC2246/5246 第 7.4.9 节，操作“服务器完成”，标签“服务器完成”。

CPA_CY_KEY_TLS_OP_USER_DEFINED
CPA_CY_KEY_TLS_OP_USER_DEFINED 自定义的自定义操作

User Defined Operation for custom labels.
标签。

enum **_CpaCyKeyHKDFOp**

File: cpa_cy_key.h

列举型别 CpaCyKeyHKDFO

TLS Operation Types
TLS 操作类型

Enumeration of the different TLS operations that can be specified in the CpaCyKeyGenHKDFOpData.
可以在 CpaCyKeyGenHKDFOpData 中指定的不同 TLS 操作的枚举。

The function **cpaCyKeyGenTls3** accelerates the TLS HKDF, which is defined as part of RFC5869 (HKDF) and RFC8446 (TLS v1.3).
该功能 cpaCyKeyGenTls3

This enumerated type defines the support HKDF operations for extraction and expansion of keying material.
此枚举类型定义了提取和扩展密钥材料的支持 HKDF 操作。

Enumerator:
枚举器:

CPA_CY_HKDF_KEY_EXTRACT — HKDF Extract operation Corresponds to RFC5869 section 2.2, step 1 "Extract"
CPA_CY_HKDF_KEY_EXTRACT HKDF 提取操作对应于 RFC5869 第 2.2 节第 1 步“提取”

CPA_CY_HKDF_KEY_EXPAND — HKDF Expand operation Corresponds to RFC5869 section 2.3, step 2 "Expand"
CPA_CY_HKDF_KEY_EXPAND HKDF 展开操作对应于 RFC5869 第 2.3 节第 2 步“展开”

CPA_CY_HKDF_KEY_EXTRACT_EXPAND — HKDF operation This performs HKDF_EXTRACT and HKDF_EXPAND in a
CPA_CY_HKDF_KEY_EXTRACT_EXPAND 此操作执行的 HKDF 操作
HKDF _摘录和 HKDF _展开

10.11 Function

	single API invocation. 单一 API 调用。
CPA_CY_HKDF_KEY_EXPAND_LABEL *CPA _ CY _ HKDF _ KEY _ EXPAND _ LABEL HKDF TLS*	HKDF Expand label operation for TLS 1.3 *1.3 的扩展标签操作* Corresponds to RFC8446 section 7.1 Key Schedule definition for HKDF-Expand-Label, which refers to HKDF-Expand defined in RFC5869. 对应于 RFC8446 第 7.1 节 HKDF-扩展-标签的关键计划定义，该定义引用了 RFC5869 中定义的 HKDF-扩展。
CPA_CY_HKDF_KEY_EXTRACT_EXPAND_LABEL *CPA _ CY _ HKDF _ KEY _ EXTRACT _ EXPAND _ LABEL HKDF 提取加展开标签操作*	HKDF Extract plus Expand label operation for TLS 1.3 Corresponds to RFC5869 section 2.2, step 1 "Extract" followed by RFC8446 section 7.1 Key Schedule definition for 对于 TLS 1.3，对应于 RFC5869 第 2.2 节第 1 步“提取”，随后是 RFC8446 第 7.1 节关键计划定义 HKDF-Expand-Label, which refers to HKDF-Expand defined in RFC5869. HKDF 扩展标签，参考 RFC5869 中定义的 HKDF 扩展。

enum **_CpaCyKeyHKDFCipherSuite**

File: cpa_cy_key.h

列举型别_CpaCyKeyHKDFCipherSuit

TLS Operation Types
TLS 操作类型

Enumeration of the different cipher suites that may be used in a TLS v1.3 operation. This value is used to infer the sizes of the key and iv sublabel.
TLS v1.3 操作中可能使用的不同密码套件的枚举。该值用于推断键和 iv 子标签的大小。

The function **cpaCyKeyGenTls3** accelerates the TLS HKDF, which is defined as part of RFC5869 (HKDF) and RFC8446 (TLS v1.3).
该功能 cpaCyKeyGenTls3

This enumerated type defines the supported cipher suites in the TLS operation that require HKDF key operations.
此枚举类型定义了需要 HKDF 密钥操作的 TLS 操作中支持的密码套件。

10.14 Function Documentation

10.15 功能文档

cpaCyKeyGenSsl (const instanceHandle, pKeyGenCb, pCallbackTag, pKeyG
const void * pGeneratedKeyBuffer
const * *
)

```
cpaCyKeyGenSsl   (const                                    instanceHandle, pKeyGenCb, pCallbackTag, pKeyG
const  void *                                              pGeneratedKeyBuffer
const * *
)
```

SSL Key Generation Function.
SSL 密钥生成函数。

This function is used for SSL key generation. It implements the key generation function defined in section
该函数用于 SSL 密钥生成。它实现了第节中定义的密钥生成功能
6.2.2 of the SSL 3.0 specification as described in
http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt.
中描述的 SSL 3.0 规范的
6.2.2http://www.mozilla.org/projects/security/pki/nss/s
sl/draft302.txt.

The input seed is taken as a flat buffer and the generated key is returned to caller in a flat destination data buffer.
输入种子作为平面缓冲区，生成的键在平面目标数据缓冲区中返回给调用者。

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] *pKeyGenCb* Pointer to callback function to be invoked when the operation is
[in]当作业为时，要叫用的回呼函式的 pKeyGenCb 指标
complete. If this is set to a NULL value the function will operate synchronously.
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] pCallbackTag 此特定调用的不透明用户数据。将原封不动地退回
in the callback.
在回调中。
[in] *pKeyGenSslOpData* Structure containing all the data needed to perform the SSL key
[in]pkeygenslopdata 结构，包含执行 SSL 金钥所需的所有资料
generation operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
生成操作。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] *pGeneratedKeyBuffer* Caller MUST allocate a sufficient buffer to hold the key generation
[out] pGeneratedKeyBuffer 调用方必须分配足够的缓冲区来保存密钥生成
output. The data pointer SHOULD be aligned on an 8-byte boundary. The length field passed in represents the size of the buffer in bytes. The value that is returned is the size of the result key in bytes. On invocation the callback function will contain this parameter in the pOut parameter.
输出。数据指针应在 8 字节边界上对齐。传入的长度字段表示缓冲区的大小，以字节为单位。返回的值是结果键的大小，以字节为单位。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_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

See also:

另请参见:

CpaCyKeyGenSslOpData, **CpaCyGenFlatBufCbFunc**
CpaCyKeyGenSslOpData, CpaCyGenFlatBufCbFunc

```
cpaCyKeyGenTls (  const                                    instanceHandle
cpaCyKeyGenTls (  const                                    instanceHandle,
                          const pKeyGenCb, void *pCallbackTag,
                          const * pKeyGenTlsOpData,
                          *pGeneratedKeyBuffer

                        )
```

TLS Key Generation Function.

TLS 密钥生成函数。

This function is used for TLS key generation. It implements the TLS PRF (Pseudo Random Function) as defined by RFC2246 (TLS v1.0) and RFC4346 (TLS v1.1).
该函数用于 TLS 密钥生成。它实现 RFC2246 (TLS v1.0)和 RFC4346 (TLS v1.1)定义的 TLS PRF(伪随机函数)。

The input seed is taken as a flat buffer and the generated key is returned to caller in a flat destination data buffer.
输入种子作为平面缓冲区，生成的键在平面目标数据缓冲区中返回给调用者。

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] *pKeyGenCb* 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，要叫用的回呼函式的 pKeyGenCb 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。将原封不动地退回

在回调中。

[in] *pKeyGenTlsOpData* Structure containing all the data needed to perform the TLS key generation 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] pKeyGenTlsOpData 结构，包含执行 TLS 金钥所需的所有资料

生成操作。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] *pGeneratedKeyBuffer* Caller MUST allocate a sufficient buffer to hold the key generation output. The data pointer SHOULD be aligned on an 8-byte boundary. The length field passed in represents the size of the buffer in bytes. The value that is returned is the size of the result key in bytes. On invocation the callback function will contain this parameter in the pOut parameter.

[out] pGeneratedKeyBuffer 调用方必须分配足够的缓冲区来保存密钥生成

输出。数据指针应在 8 字节边界上对齐。传入的长度字段表示缓冲区的大小，以字节为单位。返回的值是结果键的大小，以字节为单位。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_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

See also:
另请参见:

CpaCyKeyGenTlsOpData, **CpaCyGenFlatBufCbFunc**
CpaCyKeyGenTlsOpData, CpaCyGenFlatBufCbFunc

```
cpaCyKeyGenTls2 (  const                                    instanceHandle
cpaCyKeyGenTls2 (  const                                    instanceHandle,
                           const pKeyGenCb, void *pCallbackTag,
                           const * pKeyGenTlsOpData, hashAlgorithm, *pGeneratedKeyBuffer




                         )
```

TLS Key Generation Function version 2.
TLS 密钥生成函数版本 2。

This function is used for TLS key generation. It implements the TLS PRF (Pseudo Random Function) as defined by RFC5246 (TLS v1.2).
该函数用于 TLS 密钥生成。它实现了 RFC5246 (TLS v1.2)定义的 TLS PRF(伪随机函数)。

The input seed is taken as a flat buffer and the generated key is returned to caller in a flat destination data buffer.
输入种子作为平面缓冲区，生成的键在平面目标数据缓冲区中返回给调用者。

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]	*pKeyGenCb*	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]	*pCallbackTag*	Opaque User Data for this specific call. Will be returned unchanged in the callback.
[in]	*pKeyGenTlsOpData*	Structure containing all the data needed to perform the TLS key generation 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]	*hashAlgorithm*	

参数:

[在]	*instanceHandle*	实例句柄。
[在]	*pKeyGenCb*	操作完成时要调用的回调函数的指针。如果设置为空值，函数将同步运行。
[在]	*pCallbackTag*	此特定呼叫的不透明用户数据。将在回调中不变地返回。
[在]	*pKeyGenTlsOpData*	结构，包含执行 TLS 密钥生成操作所需的所有数据。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。
[在]	*杂凑算法*	

Specifies the hash algorithm to use. According to RFC5246, this should be "SHA-256 or a stronger standard hash function."

指定要使用的哈希算法。根据 RFC5246，这应该是“SHA-256 或更强的标准哈希函数”

[out] *pGeneratedKeyBuffer* Caller MUST allocate a sufficient buffer to hold the key generation
[out] pGeneratedKeyBuffer 调用方必须分配足够的缓冲区来保存密钥生成

output. The data pointer SHOULD be aligned on an 8-byte boundary. The length field passed in represents the size of the buffer in bytes. The value that is returned is the size of the result key in bytes. On invocation the callback function will contain this parameter in the pOut parameter.
输出。数据指针应在 8 字节边界上对齐。传入的长度字段表示缓冲区的大小，以字节为单位。返回的值是结果键的大小，以字节为单位。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_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

See also:

另请参见:

CpaCyKeyGenTlsOpData, **CpaCyGenFlatBufCbFunc**
CpaCyKeyGenTlsOpData, CpaCyGenFlatBufCbFunc

```
cpaCyKeyGenTls3 ( const                                    instanceHandle, pKeyGenCb,
cpaCyKeyGenTls3 ( const                                    instanceHandle, pKeyGenCb,
const
                          void *pCallbackTag,
                          const * pKeyGenTlsOpData, cipherSuite, *pGeneratedKeyBuffer


                        )
```

TLS Key Generation Function version 3.
TLS 密钥生成函数版本 3。

This function is used for TLS key generation. It implements the TLS HKDF (HMAC Key Derivation Function) as defined by RFC5689 (HKDF) and RFC8446 (TLS 1.3).
该函数用于 TLS 密钥生成。它实现了由 RFC5689 (HKDF)和 RFC8446 (TLS 1.3)定义的 TLS HKDF (HMAC 密钥派生函数)。

The input seed is taken as a flat buffer and the generated key is returned to caller in a flat destination data buffer.
输入种子作为平面缓冲区，生成的键在平面目标数据缓冲区中返回给调用者。

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] *pKeyGenCb* Pointer to callback function to be invoked when the operation is
[in]当作业为时，要叫用的回呼函式的 pKeyGenCb 指标
complete. If this is set to a NULL value the function will operate synchronously.
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] pCallbackTag 此特定调用的不透明用户数据。将原封不动地退回
in the callback.
在回调中。

[in] *pKeyGenTlsOpData* Structure containing all the data needed to perform the TLS key
[in] pKeyGenTlsOpData 结构，包含执行 TLS 金钥所需的所有资料
generation operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback. The memory must be pinned and contiguous, suitable for DMA operations.
生成操作。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。存储器必须是固定的和连续的，适合 DMA 操作。

[in] *hashAlgorithm* Specifies the hash algorithm to use. According to RFC5246, this
[in] hashAlgorithm 指定要使用的哈希算法。根据 RFC5246，这
should be "SHA-256 or a stronger standard hash function."
应该是“SHA-256 或更强的标准哈希函数”

[out] *pGeneratedKeyBuffer* Caller MUST allocate a sufficient buffer to hold the key generation
[out] pGeneratedKeyBuffer 调用方必须分配足够的缓冲区来保存密钥生成
output. The data pointer SHOULD be aligned on an 8-byte boundary. The length field passed in represents the size of the buffer in bytes. The value that is returned is the size of the result key in bytes. On invocation the callback function will contain this parameter in the pOut parameter.
输出。数据指针应在 8 字节边界上对齐。传入的长度字段表示缓冲区的大小，以字节为单位。返回的值是结果键的大小，以字节为单位。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_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

See also:

另请参见:

CpaCyGenFlatBufCbFunc CpaCyKeyGenHKDFOpData
CpaCyGenFlatBufCbFunc CpaCyKeyGenHKDFOpData

```
cpaCyKeyGenMgf  (const                                  instanceHandle, pKeyGenCb, pCallbackTag, pKey
cpaCyKeyGenMgf    (const                                instanceHandle, pKeyGenCb, pCallbackTag, pKey
const  void *                                           pGeneratedMaskBuffer
const * *
)
```

Mask Generation Function.

掩码生成功能。

This function implements the mask generation function MGF1 as defined by PKCS#1 v2.1, and RFC3447. The input seed is taken as a flat buffer and the generated mask is returned to caller in a flat destination data buffer.
该函数实现 PKCS#1 v2.1 和 RFC3447 定义的掩码生成函数 MGF1。输入种子作为平面缓冲区，生成的掩码在平面目标数据缓冲区中返回给调用者。

Note:
注意：

The default hash algorithm used by the MGF is SHA-1. If a different hash algorithm is preferred, then see the "extended" version of this function, **cpaCyKeyGenMgfExt**.
MGF 使用的默认哈希算法是 SHA-1。如果更喜欢不同的散列算法，那么可以查看这个函数的“扩展”版本，cpaCyKeyGenMgfExt

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] *pKeyGenCb* 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，要叫用的回呼函式的 pKeyGenCb 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。将被归还
回调中不变。

[in] *pKeyGenMgfOpData* Structure containing all the data needed to perform the MGF key generation 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] pKeyGenMgfOpData 结构，包含执行 MGF 索引键所需的所有资料
生成操作。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] *pGeneratedMaskBuffer* Caller MUST allocate a sufficient buffer to hold the generated mask. The data pointer SHOULD be aligned on an 8-byte boundary. The length field passed in represents the size of the buffer in bytes. The value that is returned is the size of the generated mask in bytes. On invocation the callback function will contain this parameter in the pOut parameter.
[out] pGeneratedMaskBuffer 调用方必须分配足够的缓冲区来保存生成的
面具。数据指针应在 8 字节边界上对齐。传入的长度字段表示缓冲区的大小，以字节为单位。返回的值是生成的掩码的大小，以字节为单位。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_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

See also:

另请参见:

CpaCyKeyGenMgfOpData, **CpaCyGenFlatBufCbFunc**
CpaCyKeyGenMgfOpData, CpaCyGenFlatBufCbFunc

```
CpaStatus
cpaCyKeyGenMgfExt (  const CpaInstanceHandle instanceHandle, pKeyGenCb, pCallbackT
const CpaCyKeyGenMgfOpDataExt *  pKeyGenMgfOpDataExt,
void *  pCallbackTag,
                              const *  pKeyGenMgfOpDataExt,
                              *pGeneratedMaskBuffer
                            )
```

Extended Mask Generation Function.
扩展掩码生成功能。

This function is used for mask generation. It differs from the "base" version of the function (**cpaCyKeyGenMgf**) in that it allows the hash function used by the Mask Generation Function to be specified.
该功能用于生成掩码。它不同于函数的“基本”版本(cpaCyKeyGenMgf

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] *pKeyGenCb* 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，要叫用的回呼函式的 pKeyGenCb 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。将被归还
回调中不变。

[in] *pKeyGenMgfOpDataExt* Structure containing all the data needed to perform the extended MGF key generation 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] pKeyGenMgfOpDataExt 结构包含执行扩展所需的所有数据
MGF 密钥生成操作。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] *pGeneratedMaskBuffer* Caller MUST allocate a sufficient buffer to hold the generated mask. The data pointer SHOULD be aligned on an 8-byte boundary. The length field passed in represents the size of the
[out] pGeneratedMaskBuffer 调用方必须分配足够的缓冲区来保存生成的
面具。数据指针应在 8 字节边界上对齐。传入的长度字段表示

buffer in bytes. The value that is returned is the size of the generated mask in bytes. On invocation the callback function will contain this parameter in the pOut parameter.

Return values:
返回值:

以字节表示的缓冲区。返回的值是生成的

掩码的大小，以字节为单位。在调用时，回调函数将在 pOut 参数中包含这个参数。

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:
This function is only used to generate a mask keys from seed material.
该函数仅用于从种子素材生成遮罩关键帧。

See also:

另请参见:
CpaCyKeyGenMgfOpData, **CpaCyGenFlatBufCbFunc**
CpaCyKeyGenMgfOpData, CpaCyGenFlatBufCbFunc

```
cpaCyKeyGenQueryStats                                   ( const
                                                          struct * pKeyGenStats
                                                        )
                                                                                instanceHandle,
```

Queries the Key and Mask generation statistics specific to an instance.
查询特定于实例的键和掩码生成统计信息。

Deprecated:
Deprecated:
As of v1.3 of the Crypto API, this function has been deprecated, replaced by
从 Crypto API 1.3 版开始，此函数已被弃用，由
cpaCyKeyGenQueryStats64().
cpaCyKeyGenQueryStats64()。

This function will query a specific instance for key and mask generation statistics. The user MUST allocate the CpaCyKeyGenStats structure and pass the reference to that into this function call. This function will write the statistic results into the passed in CpaCyKeyGenStats structure.
该函数将查询特定实例的键和掩码生成统计信息。用户必须分配 CpaCyKeyGenStats 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 CpaCyKeyGenStats 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] *pKeyGenStats* Pointer to memory into which the statistics will be written.
[out] pKeyGenStats 指向将向其中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_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:

CpaCyKeyGenStats
CpaCyKeyGenStats

```
const
cpaCyKeyGenQueryStats64 (
                                                    instanceHandle,
                                                    pKeyGenStats
                                                    *
                                                    )
```

```
const
cpaCyKeyGenQueryStats64  (                                          instanceHandle,
                                        pKeyGenStats
                                      *
                                    )
```

Queries the Key and Mask generation statistics (64-bit version) specific to an instance.
查询特定于实例的键和掩码生成统计信息(64 位版本)。

This function will query a specific instance for key and mask generation statistics. The user MUST allocate the CpaCyKeyGenStats64 structure and pass the reference to that into this function call. This function will write the statistic results into the passed in CpaCyKeyGenStats64 structure.
该函数将查询特定实例的键和掩码生成统计信息。用户必须分配 CpaCyKeyGenStats64 结构，并将对该结构的引用传递到此函数调用中。该函数将把统计结果写入传入的 CpaCyKeyGenStats64 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] *pKeyGenStats* Pointer to memory into which the statistics will be written.
[out] pKeyGenStats 指向将向其中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_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:

CpaCyKeyGenStats64
CpaCyKeyGenStats64

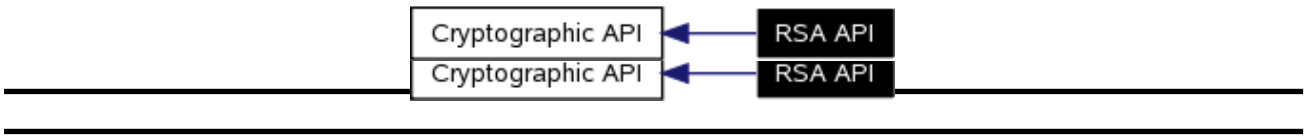
12 RSA API

13 RSA API

[Cryptographic API]

[Cryptographic API]

Collaboration diagram for RSA API:
RSA API 的协作图:



13.1 Detailed Description
13.2 详细描述

File: cpa_cy_rsa.h

文件:cpa_cy_rsa.h

These functions specify the API for Public Key Encryption (Cryptography) RSA operations. The PKCS #1 V2.1 specification is supported, however the support is limited to "two-prime" mode. RSA multi-prime is not supported.
这些函数指定了用于公钥加密(加密)RSA 操作的 API。支持 PKCS #1 V2.1 规范，但是该支持仅限于“双主”模式。不支持 RSA 多素数。

Note:
注意:

These functions implement RSA cryptographic primitives. RSA padding schemes are not implemented. For padding schemes that require the mgf function see **Cryptographic Key and Mask Generation API**.
这些函数实现 RSA 加密原语。没有实现 RSA 填充方案。有关需要 mgf 函数的填充方案，请参见 Cryptographic Key and MaskGeneration API

Large numbers are represented on the QuickAssist API as described in the Large Number API (**Cryptographic Large Number API**).
大数在 QuickAssist API 上表示，如大数 API(Cryptographic Large Number API

13.3 Data Structures
13.4 数据结构

- struct _**CpaCyRsaPublicKey**

- 结构体_CpaCyRsaPublicKey
- struct _**CpaCyRsaPrivateKeyRep1**
- 结构体_CpaCyRsaPrivateKeyRep1
- struct _**CpaCyRsaPrivateKeyRep2**
- 结构体_CpaCyRsaPrivateKeyRep2
- struct _**CpaCyRsaPrivateKey**
- 结构体_CpaCyRsaPrivateKey
- struct _**CpaCyRsaKeyGenOpData**
- 结构体_CpaCyRsaKeyGenOpData
- struct _**CpaCyRsaEncryptOpData**
- 结构体_CpaCyRsaEncryptOpData
- struct _**CpaCyRsaDecryptOpData**
- 结构体_CpaCyRsaDecryptOpData
- struct _**CpaCyRsaStats**
- 结构体_CpaCyRsaStats
- struct _**CpaCyRsaStats64**
- 结构体_CpaCyRsaStats64

13.5 Typedefs
13.6 类型定义

- typedef enum _**CpaCyRsaVersion CpaCyRsaVersion**

- typedef 枚举_CpaCyRsaVersion CpaCyRsaVersion
- typedef _**CpaCyRsaPublicKey CpaCyRsaPublicKey**
- 数据类型说明_CpaCyRsaPublicKey CpaCyRsaPublicKey
- typedef _**CpaCyRsaPrivateKeyRep1 CpaCyRsaPrivateKeyRep1**
- 数据类型说明_CpaCyRsaPrivateKeyRep1 CpaCyRsaPrivateKeyRep1
- typedef _**CpaCyRsaPrivateKeyRep2 CpaCyRsaPrivateKeyRep2**
- 数据类型说明_CpaCyRsaPrivateKeyRep2 CpaCyRsaPrivateKeyRep2
- typedef enum _**CpaCyRsaPrivateKeyRepType CpaCyRsaPrivateKeyRepType**
- typedef 枚举_CpaCyRsaPrivateKeyRepType CpaCyRsaPrivateKeyRepType
- typedef _**CpaCyRsaPrivateKey CpaCyRsaPrivateKey**
- 数据类型说明_CpaCyRsaPrivateKey CpaCyRsaPrivateKey
- typedef _**CpaCyRsaKeyGenOpData CpaCyRsaKeyGenOpData**
- 数据类型说明_CpaCyRsaKeyGenOpData CpaCyRsaKeyGenOpData
- typedef _**CpaCyRsaEncryptOpData CpaCyRsaEncryptOpData**
- 数据类型说明_CpaCyRsaEncryptOpData CpaCyRsaEncryptOpData
- typedef _**CpaCyRsaDecryptOpData CpaCyRsaDecryptOpData**
- 数据类型说明_CpaCyRsaDecryptOpData CpaCyRsaDecryptOpData
- typedef _**CpaCyRsaStats CPA_DEPRECATED**
- 数据类型说明_CpaCyRsaStats CPA_DEPRECATED
- typedef _**CpaCyRsaStats64 CpaCyRsaStats64**
- 数据类型说明_CpaCyRsaStats64 CpaCyRsaStats64
- typedef void(* **CpaCyRsaKeyGenCbFunc**)(void *pCallbackTag, **CpaStatus** status, void
- typedef void(*CpaCyRsaKeyGenCbFunc CpaStatus *pKeyGenOpData, **CpaCyRsaPrivateKey** *pPrivateKey, **CpaCyRsaPublicKey** *pPublicKey) *pKeyGenOpData, CpaCyRsaPrivateKey CpaCyRsaPublicKey

13.7 Enumerations

13.8 列举

11.4 Enumerations

11.5 列举

- enum **_CpaCyRsaVersion** { **CPA_CY_RSA_VERSION_TWO_PRIME** }

- 列举型别_CpaCyRsaVersion CPA_CY_RSA_VERSION_TWO_PRIME
- enum **_CpaCyRsaPrivateKeyRepType** {
  CPA_CY_RSA_PRIVATE_KEY_REP_TYPE_1,
  CPA_CY_RSA_PRIVATE_KEY_REP_TYPE_2
- 列举型别_CpaCyRsaPrivateKeyRepType
  CPA_CY_RSA_PRIVATE_KEY_REP_TYPE_1CPA_CY_RS
  A_PRIVATE_KEY_REP_TYPE_2
  }
  }

11.6 Functions
11.7 功能

- **CpaStatus cpaCyRsaGenKey** (const **CpaInstanceHandle** instanceHandle, const

- CpaStatus cpaCyRsaGenKey（常量CpaInstanceHandle
  CpaCyRsaKeyGenCbFunc pRsaKeyGenCb, void *pCallbackTag, const **CpaCyRsaKeyGenOpData**
  CpaCyRsaKeyGenCbFunc pRsaKeyGenCb, void *pCallbackTag, constCpaCyRsaKeyGenOpData
  *pKeyGenOpData, **CpaCyRsaPrivateKey** *pPrivateKey, **CpaCyRsaPublicKey** *pPublicKey)
  *pKeyGenOpData, CpaCyRsaPrivateKey CpaCyRsaPublicKey
- **CpaStatus cpaCyRsaEncrypt** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyRsaEncrypt（常量CpaInstanceHandle
  CpaCyGenFlatBufCbFunc pRsaEncryptCb, void *pCallbackTag, const **CpaCyRsaEncryptOpData**
  CpaCyGenFlatBufCbFunc pRsaEncryptCb, void *pCallbackTag, constCpaCyRsaEncryptOpData
  *pEncryptOpData, **CpaFlatBuffer** *pOutputData)
  *pEncryptOpData, CpaFlatBuffer
- **CpaStatus cpaCyRsaDecrypt** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyRsaDecrypt（常量CpaInstanceHandle
  CpaCyGenFlatBufCbFunc pRsaDecryptCb, void *pCallbackTag, const **CpaCyRsaDecryptOpData**
  CpaCyGenFlatBufCbFunc pRsaDecryptCb, void *pCallbackTag, constCpaCyRsaDecryptOpData
  *pDecryptOpData, **CpaFlatBuffer** *pOutputData)
  *pDecryptOpData, CpaFlatBuffer
- **CpaStatus CPA_DEPRECATED cpaCyRsaQueryStats** (const **CpaInstanceHandle**
- CpaStatus CPA_DEPRECATED cpaCyRsaQueryStats（常量CpaInstanceHandle
  instanceHandle, struct **_CpaCyRsaStats** *pRsaStats)
  instanceHandle, 结构_CpaCyRsaStats
- **CpaStatus cpaCyRsaQueryStats64** (const **CpaInstanceHandle** instanceHandle,
- CpaStatus cpaCyRsaQueryStats64（常量CpaInstanceHandle
  CpaCyRsaStats64 *pRsaStats)
  CpaCyRsaStats64 *pRsaStats)

11.8 Data Structure Documentation

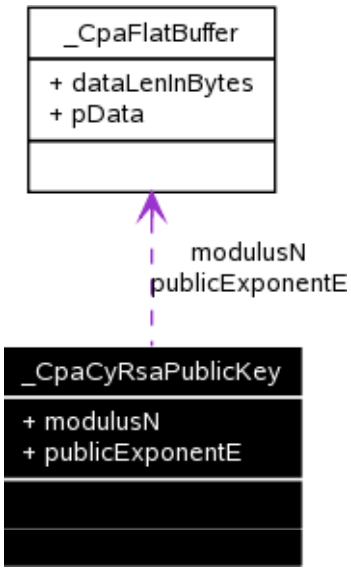
11.9 数据结构文档

11.6.1 _CpaCyRsaPublicKey Struct Reference

11.6.2 _ CpaCyRsaPublicKey 结构引用

Collaboration diagram for _CpaCyRsaPublicKey:

_ CpaCyRsaPublicKey 的协作图:



11.6.2.1 Detailed Description
11.6.2.2 详细描述

RSA Public Key Structure.
RSA 公钥结构。

This structure contains the two components which comprise the RSA public key as defined in the PKCS #1 V2.1 standard. All values in this structure are required to be in Most Significant Byte first order, e.g.
该结构包含两个组成部分，它们构成了 PKCS #1 V2.1 标准中定义的 RSA 公钥。该结构中的所有值都要求按最高有效字节优先顺序排列，例如

11.6.1 _CpaCyRsaPublicKey Struct Reference

modulusN.pData[0] = MSB.

11.6.2_ CpaCyRsaPublicKey 结构引用 module

usn . pdata[0]= MSB。

11.6.2.3 Data Fields

11.6.2.4 数据字段

- **CpaFlatBuffer modulusN**
- CpaFlatBuffer modulusN
- **CpaFlatBuffer publicExponentE**
- CpaFlatBuffer publicExponentE

11.6.2.5 Field Documentation

11.6.2.6 现场文件

CpaFlatBuffer _CpaCyRsaPublicKey::modulusN

The modulus (n). For key generation operations, the client MUST allocate the memory for this parameter; its value is generated. For encrypt operations this parameter is an input.

CpaFlatBuffe_CpaCyRsaPublicKey::modulus

CpaFlatBuffer _CpaCyRsaPublicKey::publicExponentE

The public exponent (e). For key generation operations, this field is unused. It is NOT generated by the interface; it is the responsibility of the client to set this to the same value as the corresponding parameter on

CpaFlatBuffe_CpaCyRsaPublicKey::publicExponent

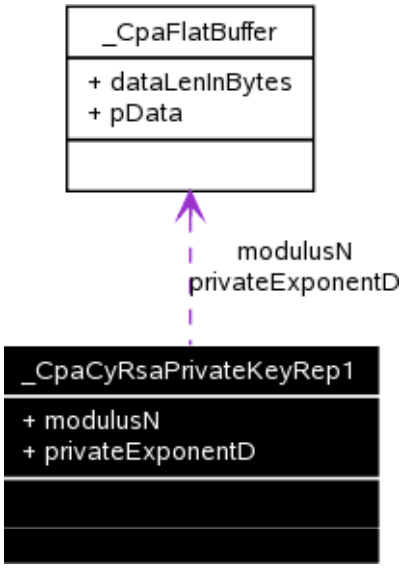
the CpaCyRsaKeyGenOpData structure before using the key for encryption. For encrypt operations this 使用密钥进行加密之前的 CpaCyRsaKeyGenOpData 结构。对于加密操作，这
parameter is an input.
参数是一个输入。

11.6.3 _CpaCyRsaPrivateKeyRep1 Struct Reference

11.6.4 _CpaCyRsaPrivateKeyRep1 结构引用

Collaboration diagram for _CpaCyRsaPrivateKeyRep1:

_CpaCyRsaPrivateKeyRep1 的协作图:



11.6.4.1 Detailed Description
11.6.4.2 详细描述

RSA Private Key Structure For Representation 1.
表示 1 的 RSA 私钥结构。

This structure contains the first representation that can be used for describing the RSA private key, represented by the tuple of the modulus (n) and the private exponent (d). All values in this structure are required to be in Most Significant Byte first order, e.g. modulusN.pData[0] = MSB.
该结构包含可用于描述 RSA 私钥的第一种表示，由模数(n)和私有指数(d)的元组表示。该结构中的所有值都要求以最高有效字节优先，例如 modulusN.pData[0] = MSB。

11.6.4.3 Data Fields
11.6.4.4 数据字段

- **CpaFlatBuffer modulusN**
- CpaFlatBuffer modulusN
- **CpaFlatBuffer privateExponentD**
- CpaFlatBuffer privateExponentD

11.6.4.5 Field Documentation

11.6.4.6 现场文件

CpaFlatBuffer _CpaCyRsaPrivateKeyRep1::modulusN

The modulus (n). For key generation operations the memory MUST be allocated by the client and the value is generated. For other operations this is an input. Permitted lengths are:

CpaFlatBuffe_CpaCyRsaPrivateKeyRep1::modulus

- 512 bits (64 bytes),
- 512 位(64 字节)，
- 1024 bits (128 bytes),
- 1024 位(128 字节)，
- 1536 bits (192 bytes),
- 1536 位(192 字节)，
- 2048 bits (256 bytes),
- 2048 位(256 字节)，
- 3072 bits (384 bytes), or
- 3072 位(384 字节)，或
- 4096 bits (512 bytes).
- 4096 位(512 字节)。

CpaFlatBuffer _CpaCyRsaPrivateKeyRep1::privateExponentD

The private exponent (d). For key generation operations the memory MUST be allocated by the client and the value is generated. For other operations this is an input. NOTE: It is important that the value D is big

CpaFlatBuffe_CpaCyRsaPrivateKeyRep1::privateExponent

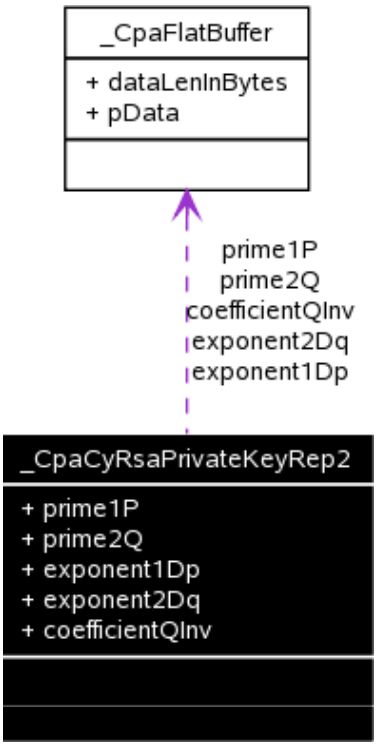
enough. It is STRONGLY recommended that this value is at least half the length of the modulus N to
够了。强烈建议该值至少为模数 N 的一半
protect against the Wiener attack.
防范香肠攻击。

11.6.5 _CpaCyRsaPrivateKeyRep2 Struct Reference

11.6.6 _CpaCyRsaPrivateKeyRep2 结构引用

Collaboration diagram for _CpaCyRsaPrivateKeyRep2:

_CpaCyRsaPrivateKeyRep2 的协作图:



11.6.6.1 Detailed Description
11.6.6.2 详细描述

RSA Private Key Structure For Representation 2.
表示 2 的 RSA 私钥结构。

This structure contains the second representation that can be used for describing the RSA private key. The quintuple of p, q, dP, dQ, and qInv (explained below and in the spec) are required for the second
此结构包含可用于描述 RSA 私钥的第二种表示形式。第二个需要五个 p、q、dP、dQ 和 qInv(在下面和规范中解释)

representation. The optional sequence of triplets are not included. All values in this structure are required to be in Most Significant Byte first order, e.g. prime1P.pData[0] = MSB.

代表性。不包括可选的三联体序列。该结构中的所有值都要求以最高有效字节优先，例如 prime1P.pData[0] = MSB。

11.6.6.3 Data Fields
11.6.6.4 数据字段

- **CpaFlatBuffer prime1P**
- CpaFlatBuffer prime1P
- **CpaFlatBuffer prime2Q**
- CpaFlatBuffer prime2Q
- **CpaFlatBuffer exponent1Dp**
- CpaFlatBuffer exponent1Dp
- **CpaFlatBuffer exponent2Dq**
- CpaFlatBuffer exponent2Dq
- **CpaFlatBuffer coefficientQInv**
- CpaFlatBuffer coefficientQInv

11.6.6.5 Field Documentation
11.6.6.6 现场文件

CpaFlatBuffer CpaCyRsaPrivateKeyRep2::prime1P

The first large prime (p). For key generation operations, this field is unused.
第一个大素数(p)。对于密钥生成操作，此字段未使用。

CpaFlatBuffer CpaCyRsaPrivateKeyRep2::prime2Q

The second large prime (q). For key generation operations, this field is unused.
第二个大素数(q)。对于密钥生成操作，此字段未使用。

CpaFlatBuffer CpaCyRsaPrivateKeyRep2::exponent1Dp

The first factor CRT exponent (dP). d mod (p-1).
第一因子 CRT 指数(dP)。d mod (p-1)。

CpaFlatBuffer CpaCyRsaPrivateKeyRep2::exponent2Dq

The second factor CRT exponent (dQ). d mod (q-1).
第二个因子 CRT 指数(dQ)。d mod (q-1)。

CpaFlatBuffer CpaCyRsaPrivateKeyRep2::coefficientQInv

The (first) Chinese Remainder Theorem (CRT) coefficient (qInv). (inverse of q) mod p.
(第一)中国剩余定理(CRT)系数(qInv)。(q 的倒数)mod p。

11.6.7 _CpaCyRsaPrivateKey Struct Reference

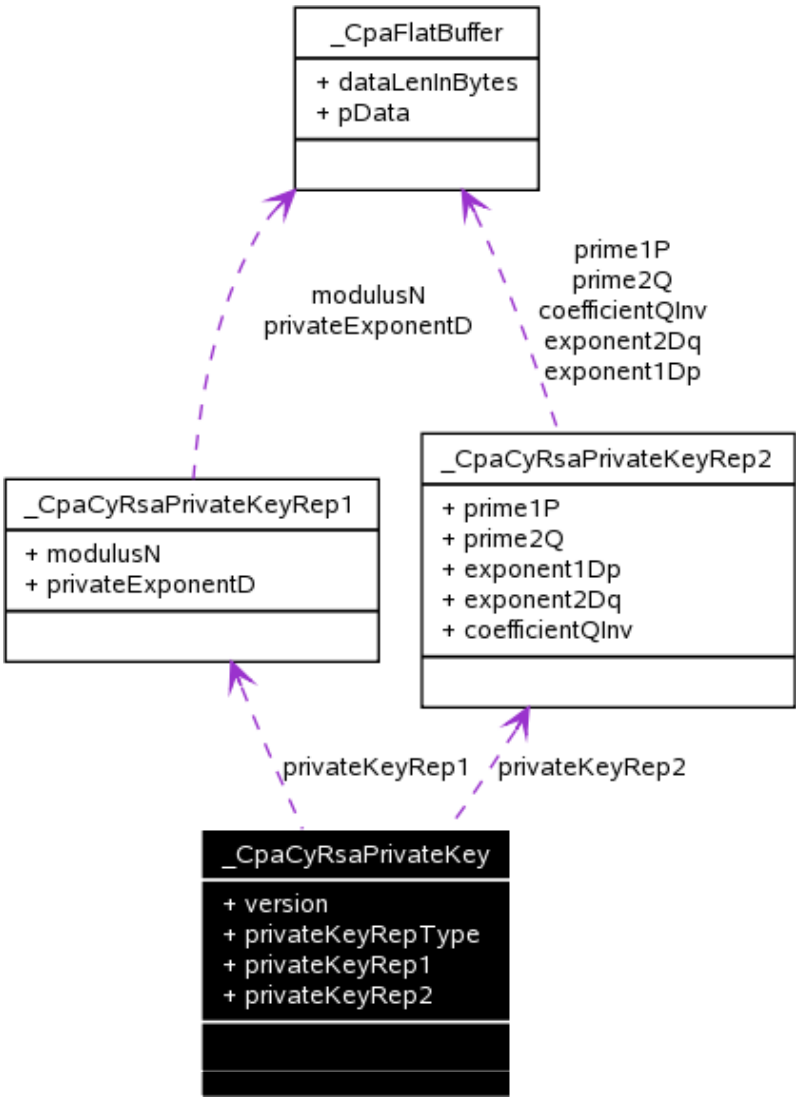
11.6.8 _CpaCyRsaPrivateKey 结构引用

Collaboration diagram for _CpaCyRsaPrivateKey:

_CpaCyRsaPrivateKey 的协作图:

11.6.4 _CpaCyRsaPrivateKey Struct Reference

11.6.5_CpaCyRsaPrivateKey 结构引用



11.6.5.1 Detailed Description
11.6.5.2 详细描述

RSA Private Key Structure.
RSA 私钥结构。

This structure contains the two representations that can be used for describing the RSA private key. The privateKeyRepType will be used to identify which representation is to be used. Typically, using the second representation results in faster decryption operations.
此结构包含可用于描述 RSA 私钥的两种表示形式。privateKeyRepType 将用于标识要使用的表示。通常，使用第二种表示会导致更快的解密操作。

11.6.5.3 Data Fields

11.6.5.4 数据字段

- **CpaCyRsaVersion version**
- CpaCyRsaVersion version
- **CpaCyRsaPrivateKeyRepType privateKeyRepType**
- CpaCyRsaPrivateKeyRepType privateKeyRepType
- **CpaCyRsaPrivateKeyRep1 privateKeyRep1**
- CpaCyRsaPrivateKeyRep1 privateKeyRep1
- **CpaCyRsaPrivateKeyRep2 privateKeyRep2**
- CpaCyRsaPrivateKeyRep2 privateKeyRep2

11.6.5.5 Field Documentation

11.6.5.6 现场文件

CpaCyRsaVersion _CpaCyRsaPrivateKey::version

CpaCyRsaVersio_CpaCyRsaPrivateKey::versio

Indicates the version of the PKCS #1 specification that is supported. Note that this applies to both representations.

表示支持的 PKCS #1 规范的版本。请注意，这适用于两种表示。

This value is used to identify which of the private key representation types in this structure is relevant. When performing key generation operations for Type 2 representations, memory must also be allocated for the type 1 representations, and values for both will be returned.

该值用于标识该结构中的哪个私钥表示类型是相关的。当对类型 2 表示执行密钥生成操作时，还必须为类型 1 表示分配内存，并且将返回两者的值。

This is the first representation of the RSA private key as defined in the PKCS #1 V2.1 specification. For key generation operations the memory for this structure is allocated by the client and the specific values are generated. For other operations this is an input parameter.

这是 PKCS #1 V2.1 规范中定义的 RSA 私钥的第一种表示形式。对于密钥生成操作，该结构的内存由客户端分配，并生成特定的值。对于其他操作，这是一个输入参数。

This is the second representation of the RSA private key as defined in the PKCS #1 V2.1 specification. For key generation operations the memory for this structure is allocated by the client and the specific values are generated. For other operations this is an input parameter.

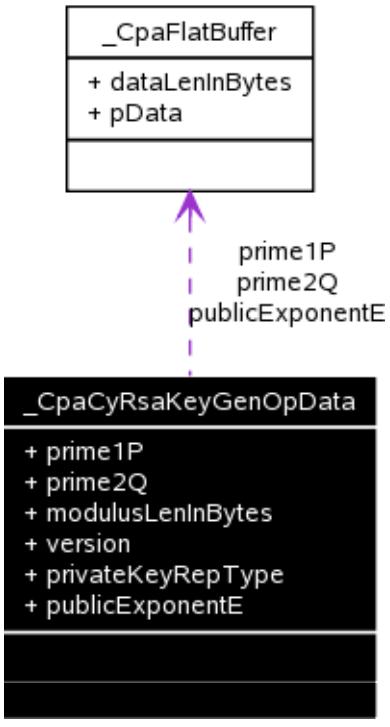
这是 PKCS #1 V2.1 规范中定义的 RSA 私钥的第二种表示形式。对于密钥生成操作，该结构的内存由客户端分配，并生成特定的值。对于其他操作，这是一个输入参数。

11.6.6 _CpaCyRsaKeyGenOpData Struct Reference

11.6.7 _CpaCyRsaKeyGenOpData 结构引用

Collaboration diagram for _CpaCyRsaKeyGenOpData:

_CpaCyRsaKeyGenOpData 的协作图:



11.6.7.1 Detailed Description
11.6.7.2 详细描述

RSA Key Generation Data.
RSA 密钥生成数据。

This structure lists the different items that are required in the cpaCyRsaGenKey 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 in the CpaCyRsaKeyGenCbFunc callback function.
此结构列出了 cpaCyRsaGenKey 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。当这个结构在 CpaCyRsaKeyGenCbFunc 回调函数中返回时，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 cpaCyRsaGenKey function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. prime1P.pData[0] = MSB.
如果客户端在将此结构中引用的内存提交给 cpaCyRsaGenKey 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。该结构中的所有值都要求以最高有效字节优先，例如 prime1P.pData[0] = MSB。

The following limitations on the permutations of the supported bit lengths of p, q and n (written as {p, q, n}) apply:

以下对 p、q 和 n(写为{p，q，n})的支持位长排列的限制适用:

- {256, 256, 512} or
- {256，256，512}或
- {512, 512, 1024} or
- {512，512，1024}或
- {768, 768, 1536} or
- {768，768，1536}或
- {1024, 1024, 2048} or
- {1024，1024，2048}或者
- {1536, 1536, 3072} or
- {1536，1536，3072}或者
- {2048, 2048, 4096}.
- {2048, 2048, 4096}.

11.6.7.3 Data Fields
11.6.7.4 数据字段

- **CpaFlatBuffer prime1P**
- CpaFlatBuffer prime1P
- **CpaFlatBuffer prime2Q**
- CpaFlatBuffer prime2Q
- **Cpa32U modulusLenInBytes**
- Cpa32U modulusLenInBytes
- **CpaCyRsaVersion version**
- CpaCyRsaVersion version
- **CpaCyRsaPrivateKeyRepType privateKeyRepType**
- CpaCyRsaPrivateKeyRepType privateKeyRepType
- **CpaFlatBuffer publicExponentE**
- CpaFlatBuffer publicExponentE

11.6.7.5 Field Documentation
11.6.7.6 现场文件

CpaFlatBuffer _CpaCyRsaKeyGenOpData::prime1P

A large random prime number (p). This MUST be created by the client. Permitted bit lengths are: 256, 512, 768, 1024, 1536 or 2048. Limitations apply - refer to the description above for details.

CpaFlatBuffe_CpaCyRsaKeyGenOpData::prime1

r

CpaFlatBuffer _CpaCyRsaKeyGenOpData::prime2Q

A large random prime number (q). This MUST be created by the client. Permitted bit lengths are: 256, 512, 768, 1024, 1536 or 2048. Limitations apply - refer to the description above for details. If the private key

CpaFlatBuffe_CpaCyRsaKeyGenOpData::prime2

representation type is 2, then this pointer will be assigned to the relevant structure member of the
表示类型为 2，则该指针将被分配给
representation 2 private key.
表示 2 私钥。

Cpa32U _CpaCyRsaKeyGenOpData::modulusLenInBytes

The bit length of the modulus (n). This is the modulus length for both the private and public keys. The length of the modulus N parameter for the private key representation 1 structure and the public key structures will

Cpa32_CpaCyRsaKeyGenOpData::modulusLenInByte

be assigned to this value. References to the strength of RSA actually refer to this bit length. Recommended
被赋予这个值。提到 RSA 的强度实际上是指这个比特长度。被推荐的
minimum is 1024 bits. Permitted lengths are:
最小值为 1024 位。允许的长度为:

- 512 bits (64 bytes),
- 512 位(64 字节)，
- 1024 bits (128 bytes),
- 1024 位(128 字节)，
- 1536 bits (192 bytes),
- 1536 位(192 字节)，
- 2048 bits (256 bytes),
- 2048 位(256 字节)，
- 3072 bits (384 bytes), or
- 3072 位(384 字节)，或
- 4096 bits (512 bytes). Limitations apply - refer to description above for details.
- 4096 位(512 字节)。限制适用-详情请参考上述说明。

CpaCyRsaVersion _CpaCyRsaKeyGenOpData::version

Indicates the version of the PKCS #1 specification that is supported. Note that this applies to both representations.

CpaCyRsaVersio_CpaCyRsaKeyGenOpData::versio

CpaCyRsaPrivateKeyRepType _CpaCyRsaKeyGenOpData::privateKeyRepType

CpaCyRsaPrivateKeyRepTyp_CpaCyRsaKeyGenOpData::privateKeyRepTyp

This value is used to identify which of the private key representation types is required to be generated.

该值用于标识需要生成哪种私钥表示类型。

CpaFlatBuffer _CpaCyRsaKeyGenOpData::publicExponentE

CpaFlatBuffe_CpaCyRsaKeyGenOpData::publicExponent

The public exponent (e).

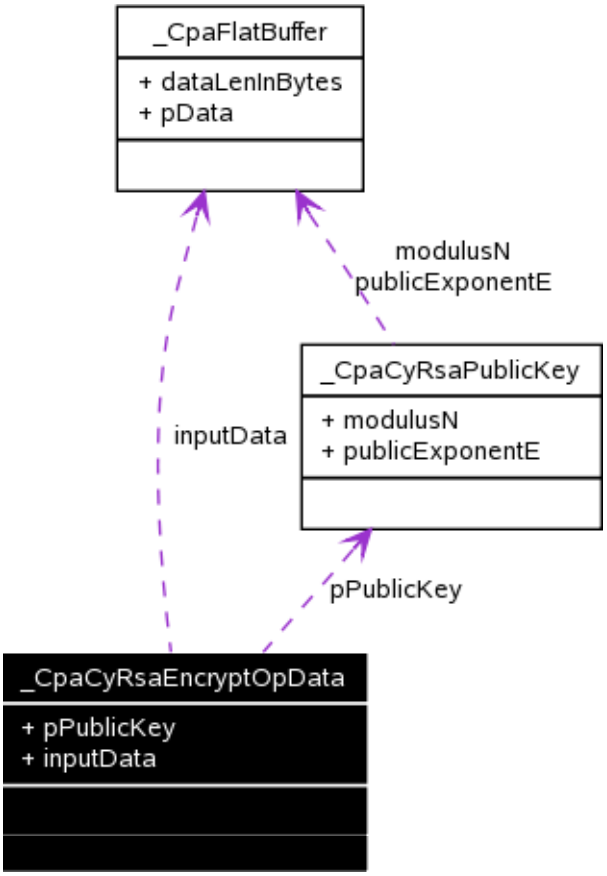
公共指数(e)。

11.6.8 _CpaCyRsaEncryptOpData Struct Reference

11.6.9 _ CpaCyRsaEncryptOpData 结构引用

Collaboration diagram for _CpaCyRsaEncryptOpData:

_ CpaCyRsaEncryptOpData 的协作图:



11.6.9.1 Detailed Description

11.6.9.2 详细描述

RSA Encryption Primitive Operation Data
RSA 加密原始操作数据

This structure lists the different items that are required in the cpaCyRsaEncrypt function. As the RSA encryption primitive and verification primitive operations are mathematically identical this structure may also be used to perform an RSA verification primitive operation. When performing an RSA encryption primitive operation, the input data is the message and the output data is the cipher text. When performing an RSA verification primitive operation, the input data is the signature and the output data is the message. 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 in the CpaCyRsaEncryptCbFunc callback function.
此结构列出了 cpaCyRsaEncrypt 函数中所需的不同项目。由于 RSA 加密原语和验证原语操作在数学上是相同的，所以该结构也可以用于执行 RSA 验证原语操作。当执行 RSA 加密原语操作时，输入数据是消息，输出数据是密文。当执行 RSA 验证原语操作时，输入数据是签名，输出数据是消息。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。当这个结构在 CpaCyRsaEncryptCbFunc 回调函数中返回时，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 cpaCyRsaEncrypt function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g.
如果客户端在将此结构中引用的内存提交给 cpaCyRsaEncrypt 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。该结构中的所有值都要求按最高有效字节优先顺序排列，例如

inputData.pData[0] = MSB.

输入数据. pData[0] = MSB。

11.6.9.3 Data Fields
11.6.9.4 数据字段

- **CpaCyRsaPublicKey * pPublicKey**
- CpaCyRsaPublicKey *pPublicKey
- **CpaFlatBuffer inputData**
- CpaFlatBuffer inputData

11.6.9.5 Field Documentation
11.6.9.6 现场文件

CpaCyRsaPublicKey* _CpaCyRsaEncryptOpData::pPublicKey
CpaCyRsaPublicKey_CpaCyRsaEncryptOpData::pPublicKe

Pointer to the public key.
指向公钥的指针。

CpaFlatBuffer _CpaCyRsaEncryptOpData::inputData

The input data that the RSA encryption primitive operation is performed on. The data pointed to is an integer that MUST be in big- endian order. The value MUST be between 0 and the modulus n - 1.

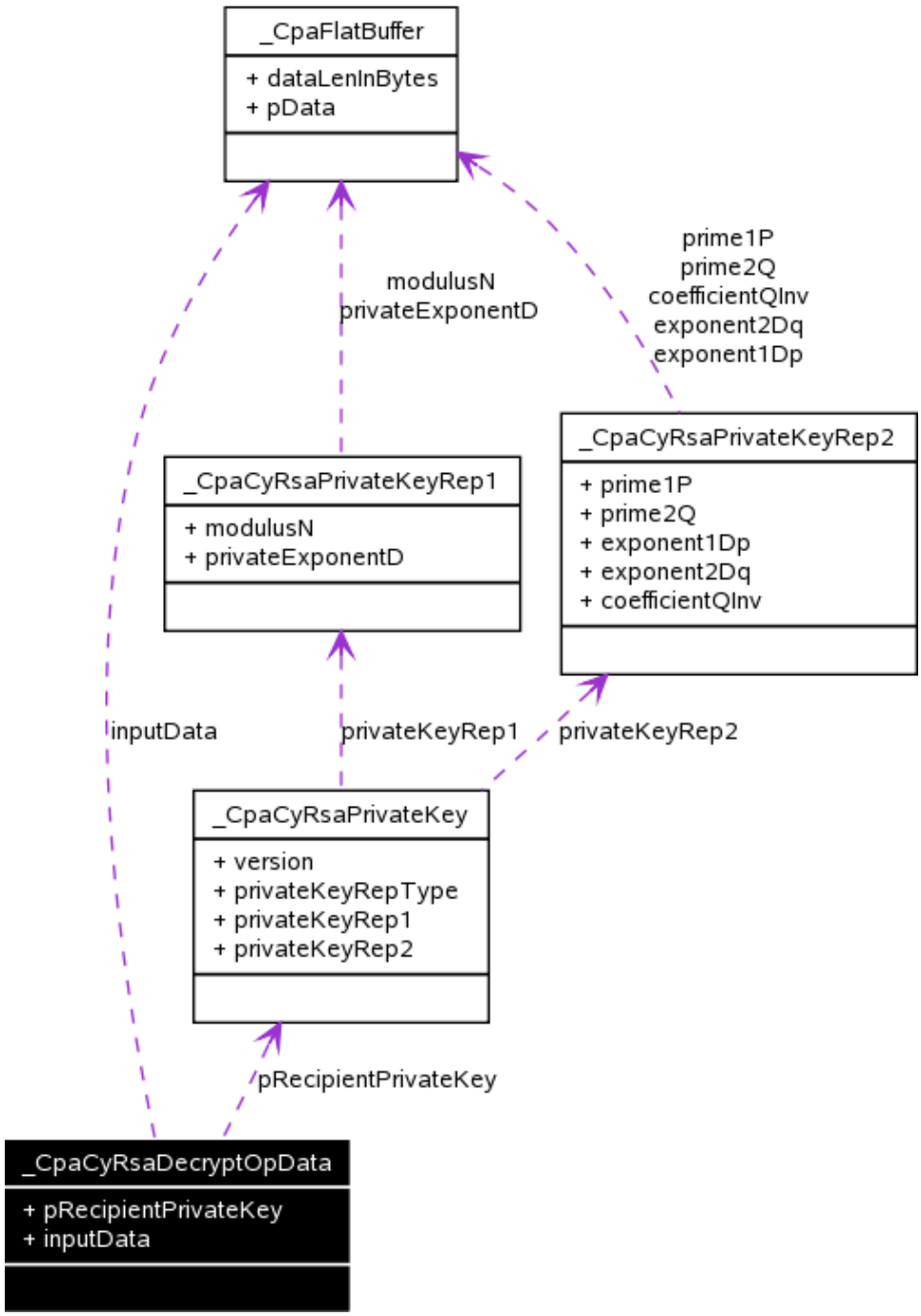
CpaFlatBuffe_CpaCyRsaEncryptOpData::inputDat

11.6.10 _CpaCyRsaDecryptOpData Struct Reference

11.6.11 _ CpaCyRsaDecryptOpData 结构引用

Collaboration diagram for _CpaCyRsaDecryptOpData:

_ CpaCyRsaDecryptOpData 的协作图:



11.6.11.1 Detailed Description

11.6.11.2 详细描述

RSA Decryption Primitive Operation Data
RSA 解密原始操作数据

This structure lists the different items that are required in the cpaCyRsaDecrypt function. As the RSA decryption primitive and signature primitive operations are mathematically identical this structure may also be used to perform an RSA signature primitive operation. When performing an RSA decryption primitive operation, the input data is the cipher text and the output data is the message text. When performing an RSA signature primitive operation, the input data is the message and the output data is the signature. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to he function. Ownership of the memory returns to the client when this structure is returned

此结构列出了 cpaCyRsaDecrypt 函数中所需的不同项目。由于 RSA 解密原语和签名原语操作在数学上是相同的，所以这种结构也可以用于执行 RSA 签名原语操作。当执行 RSA 解密原语操作时，输入数据是密文，输出数据是消息文本。当执行 RSA 签名原语操作时，输入数据是消息，输出数据是签名。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。当这个结构返回时，内存的所有权返回给客户端

in the CpaCyRsaDecryptCbFunc callback function.

在 CpaCyRsaDecryptCbFunc 回调函数中。

Note:
注意：

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyRsaDecrypt function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. inputData.pData[0] = MSB.
如果客户端在将此结构中引用的内存提交给 cpaCyRsaDecrypt 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。该结构中的所有值都要求以最高有效字节优先，例如 inputData.pData[0] = MSB。

11.6.11.3 Data Fields
11.6.11.4 数据字段

- **CpaCyRsaPrivateKey * pRecipientPrivateKey**
- CpaCyRsaPrivateKey *pRecipientPrivateKey
- **CpaFlatBuffer inputData**
- CpaFlatBuffer inputData

11.6.11.5 Field Documentation
11.6.11.6 现场文件

Pointer to the recipient's RSA private key.
指向接收者的 RSA 私钥的指针。

The input data that the RSA decryption primitive operation is performed on. The data pointed to is an integer that MUST be in big- endian order. The value MUST be between 0 and the modulus n - 1.
对其执行 RSA 解密原语操作的输入数据。指向的数据是一个必须以大端顺序排列的整数。该值必须介于 0 和模数 n - 1 之间。

11.6.12 _CpaCyRsaStats Struct Reference

11.6.13 _ CpaCyRsaStats 结构引用

11.6.13.1 Detailed Description
11.6.13.2 详细描述

RSA Statistics.
RSA 统计。

Deprecated:

Deprecated:

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by **CpaCyRsaStats64**.
从 Crypto API 的 1.3 版开始，这种结构已被取代，由 CpaCyRsaStats64

This structure contains statistics on the RSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.
此结构包含 RSA 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。

11.6.13.3 Data Fields
11.6.13.4 数据字段

- **Cpa32U numRsaKeyGenRequests**
- Cpa32U numRsaKeyGenRequests
- **Cpa32U numRsaKeyGenRequestErrors**
- Cpa32U numRsaKeyGenRequestErrors
- **Cpa32U numRsaKeyGenCompleted**
- Cpa32U numRsaKeyGenCompleted
- **Cpa32U numRsaKeyGenCompletedErrors**
- Cpa32U numRsaKeyGenCompletedErrors
- **Cpa32U numRsaEncryptRequests**
- Cpa32U numRsaEncryptRequests
- **Cpa32U numRsaEncryptRequestErrors**
- Cpa32U numRsaEncryptRequestErrors
- **Cpa32U numRsaEncryptCompleted**
- Cpa32U numRsaEncryptCompleted
- **Cpa32U numRsaEncryptCompletedErrors**
- Cpa32U numRsaEncryptCompletedErrors
- **Cpa32U numRsaDecryptRequests**
- Cpa32U numRsaDecryptRequests
- **Cpa32U numRsaDecryptRequestErrors**
- Cpa32U numRsaDecryptRequestErrors
- **Cpa32U numRsaDecryptCompleted**
- Cpa32U numRsaDecryptCompleted
- **Cpa32U numRsaDecryptCompletedErrors**
- Cpa32U numRsaDecryptCompletedErrors

11.6.13.5 Field Documentation
11.6.13.6 现场文件

11.6.8 _CpaCyRsaStats Struct Reference

11.6.9_ CpaCyRsaStats 结构引用

Total number of successful RSA key generation requests.

成功的 RSA 密钥生成请求的总数。

Total number of RSA key generation requests that had an error and could not be processed.
出现错误且无法处理的 RSA 密钥生成请求的总数。

Total number of RSA key generation operations that completed successfully.
成功完成的 RSA 密钥生成操作的总数。

Total number of RSA key generation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 RSA 密钥生成操作的总数。

Total number of successful RSA encrypt operation requests.
成功的 RSA 加密操作请求总数。

Total number of RSA encrypt requests that had an error and could not be processed.
有错误且无法处理的 RSA 加密请求总数。

Total number of RSA encrypt operations that completed successfully.
成功完成的 RSA 加密操作的总数。

Total number of RSA encrypt operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 RSA 加密操作的总数。

Total number of successful RSA decrypt operation requests.
成功的 RSA 解密操作请求的总数。

Total number of RSA decrypt requests that had an error and could not be processed.
出现错误且无法处理的 RSA 解密请求的总数。

Total number of RSA decrypt operations that completed successfully.
成功完成的 RSA 解密操作的总数。

Total number of RSA decrypt operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 RSA 解密操作的总数。

11.6.10 _CpaCyRsaStats64 Struct Reference

11.6.11 _ CpaCyRsaStats64 结构引用

11.6.11.1 Detailed Description
11.6.11.2 详细描述

RSA Statistics (64-bit version).
RSA 统计信息(64 位版本)。

This structure contains 64-bit version of the statistics on the RSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.
这个结构包含 64 位版本的 RSA 运算的统计信息。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。

11.6.11.3 Data Fields
11.6.11.4 数据字段

- **Cpa64U numRsaKeyGenRequests**
- Cpa64U numRsaKeyGenRequests
- **Cpa64U numRsaKeyGenRequestErrors**
- Cpa64U numRsaKeyGenRequestErrors
- **Cpa64U numRsaKeyGenCompleted**
- Cpa64U numRsaKeyGenCompleted
- **Cpa64U numRsaKeyGenCompletedErrors**
- Cpa64U numRsaKeyGenCompletedErrors
- **Cpa64U numRsaEncryptRequests**
- Cpa64U numRsaEncryptRequests

11.6.9 _CpaCyRsaStats64 Struct Reference

11 . 6 . 9 _ cpacyrsats 64结构引用

- **Cpa64U numRsaEncryptRequestErrors**

- Cpa64U numRsaEncryptRequestErrors
- **Cpa64U numRsaEncryptCompleted**
- Cpa64U numRsaEncryptCompleted
- **Cpa64U numRsaEncryptCompletedErrors**
- Cpa64U numRsaEncryptCompletedErrors
- **Cpa64U numRsaDecryptRequests**
- Cpa64U numRsaDecryptRequests
- **Cpa64U numRsaDecryptRequestErrors**
- Cpa64U numRsaDecryptRequestErrors
- **Cpa64U numRsaDecryptCompleted**
- Cpa64U numRsaDecryptCompleted
- **Cpa64U numRsaDecryptCompletedErrors**
- Cpa64U numRsaDecryptCompletedErrors

11.6.11.5 Field Documentation
11.6.11.6 现场文件

Total number of successful RSA key generation requests.
成功的 RSA 密钥生成请求的总数。

Total number of RSA key generation requests that had an error and could not be processed.
出现错误且无法处理的 RSA 密钥生成请求的总数。

Total number of RSA key generation operations that completed successfully.
成功完成的 RSA 密钥生成操作的总数。

Total number of RSA key generation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 RSA 密钥生成操作的总数。

Total number of successful RSA encrypt operation requests.
成功的 RSA 加密操作请求总数。

Total number of RSA encrypt requests that had an error and could not be processed.
有错误且无法处理的 RSA 加密请求总数。

Total number of RSA encrypt operations that completed successfully.
成功完成的 RSA 加密操作的总数。

Total number of RSA encrypt operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 RSA 加密操作的总数。

Total number of successful RSA decrypt operation requests.
成功的 RSA 解密操作请求的总数。

Total number of RSA decrypt requests that had an error and could not be processed.
出现错误且无法处理的 RSA 解密请求的总数。

Total number of RSA decrypt operations that completed successfully.
成功完成的 RSA 解密操作的总数。

Total number of RSA decrypt operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 RSA 解密操作的总数。

11.7 Typedef Documentation

11.8 Typedef 文档

RSA Version.

RSA 版本。

This enumeration lists the version identifier for the PKCS #1 V2.1 standard.
此枚举列出了 PKCS #1 V2.1 标准的版本标识符。

Note:
注意:

Multi-prime (more than two primes) is not supported.
不支持多素数(两个以上的素数)。

typedef struct _**CpaCyRsaPublicKey CpaCyRsaPublicKey**
typedef 结构_CpaCyRsaPublicKeCpaCyRsaPublicKe

RSA Public Key Structure.
RSA 公钥结构。

This structure contains the two components which comprise the RSA public key as defined in the PKCS #1 V2.1 standard. All values in this structure are required to be in Most Significant Byte first order, e.g. modulusN.pData[0] = MSB.
该结构包含两个组成部分，它们构成了 PKCS #1 V2.1 标准中定义的 RSA 公钥。该结构中的所有值都要求以最高有效字节优先，例如 modulusN.pData[0] = MSB。

typedef struct _**CpaCyRsaPrivateKeyRep1 CpaCyRsaPrivateKeyRep1**
typedef 结构_CpaCyRsaPrivateKeyRepCpaCyRsaPrivateKeyRep

RSA Private Key Structure For Representation 1.
表示 1 的 RSA 私钥结构。

This structure contains the first representation that can be used for describing the RSA private key, represented by the tuple of the modulus (n) and the private exponent (d). All values in this structure are required to be in Most Significant Byte first order, e.g. modulusN.pData[0] = MSB.
该结构包含可用于描述 RSA 私钥的第一种表示，由模数(n)和私有指数(d)的元组表示。该结构中的所有值都要求以最高有效字节优先，例如 modulusN.pData[0] = MSB。

typedef struct _**CpaCyRsaPrivateKeyRep2 CpaCyRsaPrivateKeyRep2**
typedef 结构_CpaCyRsaPrivateKeyRepCpaCyRsaPrivateKeyRep

RSA Private Key Structure For Representation 2.
表示 2 的 RSA 私钥结构。

This structure contains the second representation that can be used for describing the RSA private key. The quintuple of p, q, dP, dQ, and qInv (explained below and in the spec) are required for the second representation. The optional sequence of triplets are not included. All values in this structure are required to be in Most Significant Byte first order, e.g. prime1P.pData[0] = MSB.
此结构包含可用于描述 RSA 私钥的第二种表示形式。第二种表示需要 p、q、dP、dQ 和 qInv 的五元组(在下面和规范中解释)。不包括可选的三联体序列。该结构中的所有值都要求以最高有效字节优先，例如 prime1P.pData[0] = MSB。

typedef enum _CpaCyRsaPrivateKeyRepType CpaCyRsaPrivateKeyRepType

typedef 枚举_CpaCyRsaPrivateKeyRepTypCpaCyRsaPrivateKeyRepTyp

RSA private key representation type.

RSA 私钥表示类型。

This enumeration lists which PKCS V2.1 representation of the private key is being used.

此枚举列出了正在使用的私钥的 PKCS 2.1 版表示形式。

typedef struct _CpaCyRsaPrivateKey CpaCyRsaPrivateKey

typedef 结构_CpaCyRsaPrivateKeCpaCyRsaPrivateKe

RSA Private Key Structure.

RSA 私钥结构。

This structure contains the two representations that can be used for describing the RSA private key. The privateKeyRepType will be used to identify which representation is to be used. Typically, using the second representation results in faster decryption operations.

此结构包含可用于描述 RSA 私钥的两种表示形式。privateKeyRepType 将用于标识要使用的表示。通常，使用第二种表示会导致更快的解密操作。

typedef struct _CpaCyRsaKeyGenOpData CpaCyRsaKeyGenOpData

typedef 结构_CpaCyRsaKeyGenOpDatCpaCyRsaKeyGenOpDat

RSA Key Generation Data.

RSA 密钥生成数据。

This structure lists the different items that are required in the cpaCyRsaGenKey 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 in the CpaCyRsaKeyGenCbFunc callback function.

此结构列出了 cpaCyRsaGenKey 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。当这个结构在 CpaCyRsaKeyGenCbFunc 回调函数中返回时，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 cpaCyRsaGenKey function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. prime1P.pData[0] = MSB.

如果客户端在将此结构中引用的内存提交给 cpaCyRsaGenKey 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。该结构中的所有值都要求以最高有效字节优先，例如 prime1P.pData[0] = MSB。

The following limitations on the permutations of the supported bit lengths of p, q and n (written as {p, q, n}) apply:

以下对 p、q 和 n(写为{p，q，n})的支持位长排列的限制适用:

- {256, 256, 512} or
- {256，256，512}或
- {512, 512, 1024} or
- {512，512，1024}或
- {768, 768, 1536} or
- {768，768，1536}或
- {1024, 1024, 2048} or
- {1024，1024，2048}或者
- {1536, 1536, 3072} or
- {1536，1536，3072}或者
- {2048, 2048, 4096}.
- {2048, 2048, 4096}.

typedef struct _**CpaCyRsaEncryptOpData CpaCyRsaEncryptOpData**

typedef 结构_CpaCyRsaEncryptOpDatCpaCyRsaEncryptOpDat

RSA Encryption Primitive Operation Data

RSA 加密原始操作数据

This structure lists the different items that are required in the cpaCyRsaEncrypt function. As the RSA encryption primitive and verification primitive operations are mathematically identical this structure may also be used to perform an RSA verification primitive operation. When performing an RSA encryption primitive operation, the input data is the message and the output data is the cipher text. When performing an RSA verification primitive operation, the input data is the signature and the output data is the message. 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 in the CpaCyRsaEncryptCbFunc callback function.

此结构列出了 cpaCyRsaEncrypt 函数中所需的不同项目。由于 RSA 加密原语和验证原语操作在数学上是相同的，所以该结构也可以用于执行 RSA 验证原语操作。当执行 RSA 加密原语操作时，输入数据是消息，输出数据是密文。当执行 RSA 验证原语操作时，输入数据是签名，输出数据是消息。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。当这个结构在 CpaCyRsaEncryptCbFunc 回调函数中返回时，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 cpaCyRsaEncrypt function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. inputData.pData[0] = MSB.

如果客户端在将此结构中引用的内存提交给 cpaCyRsaEncrypt 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。该结构中的所有值都要求以最高有效字节优先，例如 inputData.pData[0] = MSB。

typedef struct _**CpaCyRsaDecryptOpData CpaCyRsaDecryptOpData**

typedef 结构_CpaCyRsaDecryptOpDatCpaCyRsaDecryptOpDat

RSA Decryption Primitive Operation Data

RSA 解密原始操作数据

This structure lists the different items that are required in the cpaCyRsaDecrypt function. As the RSA decryption primitive and signature primitive operations are mathematically identical this structure may also be used to perform an RSA signature primitive operation. When performing an RSA decryption primitive operation, the input data is the cipher text and the output data is the message text. When performing an RSA signature primitive operation, the input data is the message and the output data is the signature. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to he function. Ownership of the memory returns to the client when this structure is returned in the CpaCyRsaDecryptCbFunc callback function.
此结构列出了 cpaCyRsaDecrypt 函数中所需的不同项目。由于 RSA 解密原语和签名原语操作在数学上是相同的，所以这种结构也可以用于执行 RSA 签名原语操作。当执行 RSA 解密原语操作时，输入数据是密文，输出数据是消息文本。当执行 RSA 签名原语操作时，输入数据是消息，输出数据是签名。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。当此结构在 CpaCyRsaDecryptCbFunc 回调函数中返回时，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 cpaCyRsaDecrypt function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. inputData.pData[0] = MSB.
如果客户端在将此结构中引用的内存提交给 cpaCyRsaDecrypt 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。该结构中的所有值都要求以最高有效字节优先，例如 inputData.pData[0] = MSB。

typedef struct _**CpaCyRsaStats CPA_DEPRECATED**
typedef 结构_CpaCyRsaStatCPA_DEPRECATE

RSA Statistics.
RSA 统计。

Deprecated:
Deprecated:

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by **CpaCyRsaStats64**.
从 Crypto API 的 1.3 版开始，这种结构已被取代，由 CpaCyRsaStats64

This structure contains statistics on the RSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.
此结构包含 RSA 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 _CpaCyRsaStats64 CpaCyRsaStats64

typedef 结构_CpaCyRsaStats6CpaCyRsaStats6

RSA Statistics (64-bit version).

RSA 统计信息(64 位版本)。

This structure contains 64-bit version of the statistics on the RSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.

这个结构包含 64 位版本的 RSA 运算的统计信息。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 void(* CpaCyRsaKeyGenCbFunc)(void *pCallbackTag, CpaStatus status, void *pKeyGenOp

Definition of the RSA key generation callback function.

RSA 密钥生成回调函数的定义。

typedef void(* CpaCyRsaKeyGenCbFunc)(void *pCallbackTag, CpaStatus status, void *pKeyGenOp

This is the prototype for the RSA key generation callback function. The callback function pointer is passed in as a parameter to the cpaCyRsaGenKey function. It will be invoked once the request has completed.

这是 RSA 密钥生成回调函数的原型。回调函数指针作为参数传递给 cpaCyRsaGenKey 函数。一旦请求完成，它将被调用。

Context:

背景:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.

这个回调函数可以在不允许休眠发生的上下文中执行。

Assumptions:

假设:

None

没有人

Side-Effects:

副作用:

None

没有人

Reentrant:

可重入:

No

不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

[in] *pCallbackTag* Opaque value provided by user while making individual function calls.

[in] pCallbackTag 使用者在进行个别函式呼叫时所提供的不透明值。

[in] *status* Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.

[in] status 操作的状态。有效值为 CPA_STATUS_SUCCESS、CPA_STATUS_FAIL 和 CPA_STATUS_UNSUPPORTED。

[in] *pKeyGenOpData* Structure with output params for callback.

[in]具有回调输出参数的 pKeyGenOpData 结构。

[in] *pPrivateKey* Structure which contains pointers to the memory into which the generated private key will be written.

[in] pPrivateKey 结构，包含指向将写入生成的私钥的内存的指针。

[in] *pPublicKey* Structure which contains pointers to the memory into which the generated public key will be written. The pointer to the public exponent (e) that is returned in this structure is equal to the input public exponent.

[in] pPublicKey 结构，包含指向生成的公钥将写入其中的内存的指针。此结构中返回的公共指数 (e)的指针等于输入公共指数。

Return values:

返回值:

None

没有人

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:

CpaCyRsaPrivateKey, **CpaCyRsaPublicKey**, **cpaCyRsaGenKey()**

CpaCyRsaPrivateKey, CpaCyRsaPublicKeycpaCyRsaGenKey()

11.8 Enumeration Type Documentation

11.9 枚举类型文档

RSA Version. enum **CpaCyRsaVersion**
RSA 版本。

This enumeration lists the version identifier for the PKCS #1 V2.1 standard.
此枚举列出了 PKCS #1 V2.1 标准的版本标识符。

Note:
注意:

Multi-prime (more than two primes) is not supported.
不支持多素数(两个以上的素数)。

Enumerator:
枚举器:

CPA_CY_RSA_VERSION_TWO_PRIME The version supported is "two-prime".
CPA_CY_RSA_VERSION_TWO_PRIME 支持的版本是“两撇”。

RSA private key representation type. enum **CpaCyRsaPrivateKeyRepType**
RSA 私钥表示类型。

This enumeration lists which PKCS V2.1 representation of the private key is being used.
此枚举列出了正在使用的私钥的 PKCS 2.1 版表示形式。

Enumerator:
枚举器:

CPA_CY_RSA_PRIVATE_KEY_REP_TYPE_1 The first representation of the RSA private key.
*CPA_CY_RSA_PRIVATE_KEY_REP_TYPE_1 RSA PRIVATE 的第一个表示*钥匙。

CPA_CY_RSA_PRIVATE_KEY_REP_TYPE_2 The second representation of the RSA private key.
*CPA_CY_RSA_PRIVATE_KEY_REP_TYPE_2 RSA 的第二种表示*私钥。

11.10 Function Documentation

11.11 功能文档

```
cpaCyRsaGenKey (const                    instanceHandle, pRsaKeyGenCb, pCallbackTag, p
const void *                             pPrivateKey, pPublicKey
const * * *
)
```

```
cpaCyRsaGenKey    (const                                    instanceHandle, pRsaKeyGenCb, pCallbackTag, p
const  void *                                               pPrivateKey, pPublicKey
const * * *
)
```

Generate RSA keys.
生成 RSA 密钥。

This function will generate private and public keys for RSA as specified in the PKCS #1 V2.1 standard. Both representation types of the private key may be generated.
该函数将按照 PKCS #1 V2.1 标准的规定为 RSA 生成私钥和公钥。可以生成私钥的两种表示类型。

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] *pRsaKeyGenCb* Pointer to the callback function to be invoked when the operation is complete. If this is set to a NULL value the function will operate synchronously.
[in] pRsaKeyGenCb 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] *pKeyGenOpData* Structure containing all the data needed to perform the RSA key generation 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] pKeyGenOpData 结构，包含执行 RSA 金钥产生作业所需的所有资料。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] *pPrivateKey* Structure which contains pointers to the memory into which the generated private key will be written. The client MUST allocate memory for this structure, and for the pointers within it, recursively; on return, these will be populated.
[out] pPrivateKey 结构，包含产生的私密金钥将会写入的记忆体指标。客户端必须为这个结构以及其中的指针递归地分配内存；返回时，这些将被填充。

[out] *pPublicKey* Structure which contains pointers to the memory into which the generated public key will be written. The memory for this structure and for the modulusN parameter MUST be allocated by the client, and will be populated on return from the call. The field publicExponentE is not modified or touched in any way; it is the responsibility of the client to set this to the same value as the corresponding parameter on the CpaCyRsaKeyGenOpData structure before using the key for encryption.
[out] pPublicKey 结构，包含产生的公开金钥将会写入的记忆体指标。此结构和 modulusN 参数的内存必须由客户端分配，并将在调用返回时填充。publicExponentE 字段不会以任何方式被修改或更改；在使用密钥进行加密之前，客户端负责将其设置为与 CpaCyRsaKeyGenOpData 结构上的相应参数相同的值。

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 pRsaKeyGenCb	is non-NULL, an asynchronous callback of type 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. 当 pRsaKeyGenCb 为非 NULL 时，将生成一个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。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:

CpaCyRsaKeyGenOpData, **CpaCyRsaKeyGenCbFunc**, **cpaCyRsaEncrypt()**, **cpaCyRsaDecrypt()**

CpaCyRsaKeyGenOpData, CpaCyRsaKeyGenCbFunccpaCyRsaEncrypt () cpaCyRsaDecrypt ()

cpaCyRsaEncrypt (const const void * const * *)	instanceHandle, pRsaEncryptCb, pCallbackTag, pOutputData
cpaCyRsaEncrypt (const const void * const * *)	instanceHandle, pRsaEncryptCb, pCallbackTag pOutputData

Perform the RSA encrypt (or verify) primitive operation on the input data.
对输入数据执行 RSA 加密(或验证)原语操作。

This function will perform an RSA encryption primitive operation on the input data using the specified RSA public key. As the RSA encryption primitive and verification primitive operations are mathematically identical this function may also be used to perform an RSA verification primitive operation.
该函数将使用指定的 RSA 公钥对输入数据执行 RSA 加密原语操作。由于 RSA 加密原语和验证原语操作在数学上是相同的，因此该函数也可用于执行 RSA 验证原语操作。

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] *pRsaEncryptCb* 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] pRsaEncryptCb 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] *pEncryptOpData* Structure containing all the data needed to perform the RSA encryption 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] pEncryptOpData 结构，包含执行 RSA 加密作业所需的所有资料。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 encryption 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 加密基元的结果所写入的结构的指针。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。

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.
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.

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 pRsaEncryptCb	is non-NULL an asynchronous callback of type 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. 当 pRsaEncryptCb 为非 NULL 时，将生成类型的异步回调以响应此函数调用。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。

See also:

另请参见:

CpaCyGenFlatBufCbFunc CpaCyRsaEncryptOpData cpaCyRsaGenKey() cpaCyRsaDecrypt()

CpaCyGenFlatBufCbFunc CpaCyRsaEncryptOpData cpaCyRsaGenKey() cpaCyRsaDecrypt()

```
cpaCyRsaDecrypt (const                                  instanceHandle, pRsaDecryptCb, pCallbackTag, p
cpaCyRsaDecrypt    (const
const  void *
const * *
)
                                                        instanceHandle, pRsaDecryptCb, pCallbackTag,
                                                        pOutputData
```

Perform the RSA decrypt (or sign) primitive operation on the input data.
对输入数据执行 RSA 解密(或签名)原语操作。

This function will perform an RSA decryption primitive operation on the input data using the specified RSA private key. 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.
此函数将使用指定的 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] pRsaDecryptCb 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] *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 解密作业所需的所有资料。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 解密基元的结果所写入的结构的指针。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。

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 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.
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。

See also:

另请参见：

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

Query statistics for a specific RSA instance.
查询特定 RSA 实例的统计信息。

CpaStatus CPA_DEPRECATED cpaCyRsaQueryStats (const **CpaInstanceHandle**
CpaStatus CPA_DEPRECATED cpaCyRsaQueryStats(const CpaInstanceHandle *instanceHandle*,
struct **CpaCyRsaStats** * *pRsaStats*

Deprecated:
Deprecated:

As of v1.3 of the Crypto API, this function has been deprecated, replaced by
从 Crypto API 1.3 版开始，此函数已被弃用，由
cpaCyRsaQueryStats64().
cpaCyRsaQueryStats64()。

This function will query a specific instance for RSA statistics. The user MUST allocate the CpaCyRsaStats structure and pass the reference to that into this function call. This function will write the statistic results into the passed in CpaCyRsaStats structure.
该函数将查询特定实例的 RSA 统计信息。用户必须分配 CpaCyRsaStats 结构，并将对该结构的引用传递到这个函数调用中。该函数将把统计结果写入传入的 CpaCyRsaStats 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] *pRsaStats* Pointer to memory into which the statistics will be written.
[out]pr stats 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_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。

CPA_STATUS_UNSUPPORTED Function is not supported.
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:
CpaCyRsaStats
CpaCyRsaStats

Query statistics (64-bit version) for a specific RSA instance.
特定 RSA 实例的查询统计信息(64 位版本)。

```
CpaStatus cpaCyRsaQueryStats64( const CpaInstanceHandle
CpaStatus cpaCyRsaQueryStats64( const CpaInstanceHandle
    instanceHandle,
    CpaCyRsaStats64 *          pRsaStats
    CpaCyRsaStats64 *          pRsaStats
```

This function will query a specific instance for RSA statistics. The user MUST allocate the CpaCyRsaStats64 structure and pass the reference to that into this function call. This function will write the statistic results into the passed in CpaCyRsaStats64 structure.
该函数将查询特定实例的 RSA 统计信息。用户必须分配 CpaCyRsaStats64 结构，并将对该结构的引用传递到此函数调用中。该函数将把统计结果写入传入的 CpaCyRsaStats64 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] *pRsaStats* Pointer to memory into which the statistics will be written.
[out]pr stats 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_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。

CPA_STATUS_UNSUPPORTED Function is not supported.
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:
CpaCyRsaStats64
CpaCyRsaStats64

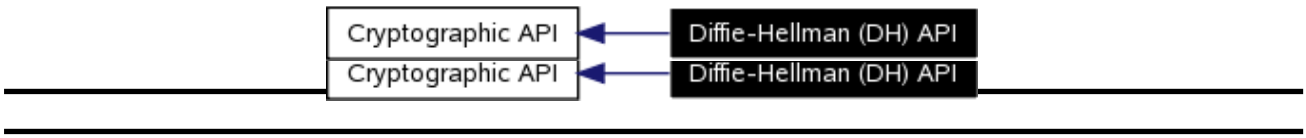
14 Diffie-Hellman (DH) API

15 迪菲-赫尔曼(DH) API

[Cryptographic API]

[Cryptographic API]

Collaboration diagram for Diffie-Hellman (DH) API:
Diffie-Hellman (DH) API 的协作图:



15.1 Detailed Description

15.2 详细描述

File: cpa_cy_dh.h

文件:cpa_cy_dh.h

These functions specify the API for Public Key Encryption (Cryptography) operations for use with Diffie-Hellman algorithm.
这些函数指定了用于 Diffie-Hellman 算法的公钥加密(加密)操作的 API。

Note:
注意:

Large numbers are represented on the QuickAssist API as described in the Large Number API (**Cryptographic Large Number API**).
大数在 QuickAssist API 上表示，如大数 API(Cryptographic Large Number API

15.3 Data Structures

15.4 数据结构

- struct _**CpaCyDhPhase1KeyGenOpData**
- 结构体_CpaCyDhPhase1KeyGenOpData
- struct _**CpaCyDhPhase2SecretKeyGenOpData**
- 结构体_CpaCyDhPhase2SecretKeyGenOpData
- struct _**CpaCyDhStats**
- 结构体_CpaCyDhStats
- struct _**CpaCyDhStats64**

- 结构体_CpaCyDhStats64

15.5 Typedefs
15.6 类型定义

- typedef _**CpaCyDhPhase1KeyGenOpData CpaCyDhPhase1KeyGenOpData**

- 数据类型说明_CpaCyDhPhase1KeyGenOpData CpaCyDhPhase1KeyGenOpData
- typedef _**CpaCyDhPhase2SecretKeyGenOpData CpaCyDhPhase2SecretKeyGenOpData**
- 数据类型说明_CpaCyDhPhase2SecretKeyGenOpData CpaCyDhPhase2SecretKeyGenOpData
- typedef _**CpaCyDhStats CPA_DEPRECATED**
- 数据类型说明_CpaCyDhStats CPA_DEPRECATED
- typedef _**CpaCyDhStats64 CpaCyDhStats64**
- 数据类型说明_CpaCyDhStats64 CpaCyDhStats64

15.7 Functions
15.8 功能

- **CpaStatus cpaCyDhKeyGenPhase1** (const **CpaInstanceHandle** instanceHandle, const **CpaCyGenFlatBufCbFunc** pDhPhase1Cb, void *pCallbackTag, const **CpaCyDhPhase1KeyGenOpData** *pPhase1KeyGenData, **CpaFlatBuffer** *pLocalOctetStringPV)

- CpaStatus cpaCyDhKeyGenPhase1（常量CpaInstanceHandle CpaCyGenFlatBufCbFunc CpaCyDhPhase1KeyGenOpData CpaFlatBuffer
- **CpaStatus cpaCyDhKeyGenPhase2Secret** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyDhKeyGenPhase2Secret（常量CpaInstanceHandle **CpaCyGenFlatBufCbFunc** pDhPhase2Cb, void *pCallbackTag, const CpaCyGenFlatBufCbFunc pDhPhase2Cb, void *pCallbackTag, const **CpaCyDhPhase2SecretKeyGenOpData** *pPhase2SecretKeyGenData, **CpaFlatBuffer** CpaCyDhPhase2SecretKeyGenOpData *pPhase2SecretKeyGenData, CpaFlatBuffer *pOctetStringSecretKey) *pOctetStringSecretKey)
- **CpaStatus CPA_DEPRECATED cpaCyDhQueryStats** (const **CpaInstanceHandle** instanceHandle, struct _**CpaCyDhStats** *pDhStats)
- CpaStatus CPA_DEPRECATED cpaCyDhQueryStats（常量CpaInstanceHandle _CpaCyDhStats
- **CpaStatus cpaCyDhQueryStats64** (const **CpaInstanceHandle** instanceHandle, **CpaCyDhStats64**
- CpaStatus cpaCyDhQueryStats64（常量CpaInstanceHandle CpaCyDhStats64 *pDhStats) *pDhStats)

15.9 Data Structure Documentation

15.10 数据结构文档

12.5 Data Structure Documentation

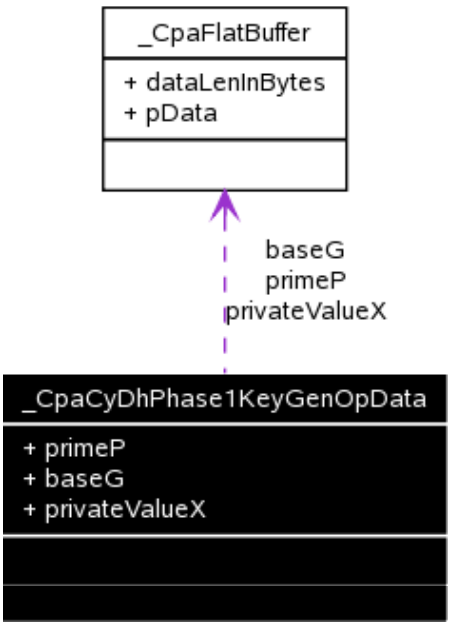
12.6 数据结构文档

12.6.1 _CpaCyDhPhase1KeyGenOpData Struct Reference

12.6.2 _CpaCyDhPhase1KeyGenOpData 结构引用

Collaboration diagram for _CpaCyDhPhase1KeyGenOpData:

_CpaCyDhPhase1KeyGenOpData 的协作图:



12.6.2.1 Detailed Description
12.6.2.2 详细描述

Diffie-Hellman Phase 1 Key Generation Data.
Diffie-Hellman 第一阶段密钥生成数据。

This structure lists the different items that are required in the cpaCyDhKeyGenPhase1 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 CpaCyDhPhase1KeyGenOpData structure.
此结构列出了 cpaCyDhKeyGenPhase1 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。当这个结构与 CpaCyDhPhase1KeyGenOpData 结构一起返回时，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 cpaCyDhKeyGenPhase1 function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. primeP.pData[0] = MSB.
如果客户端在将此结构中引用的内存提交给 cpaCyDhKeyGenPhase1 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。该结构中的所有值都要求以最高有效字节优先，例如 primeP.pData[0] = MSB。

12.6.2.3 Data Fields
12.6.2.4 数据字段

- **CpaFlatBuffer primeP**
- CpaFlatBuffer primeP
- **CpaFlatBuffer baseG**
- CpaFlatBuffer baseG
- **CpaFlatBuffer privateValueX**
- CpaFlatBuffer privateValueX

12.6.2.5 Field Documentation
12.6.2.6 现场文件

CpaFlatBuffer _CpaCyDhPhase1KeyGenOpData::primeP

Flat buffer containing a pointer to the random odd prime number (p). The bit-length of this number may be one of 768, 1024, 1536, 2048, 3072 or 4096.

CpaFlatBuffe_CpaCyDhPhase1KeyGenOpData::prime

12.5.1 _CpaCyDhPhase1KeyGenOpData Struct Reference

12.5.2_CpaCyDhPhase1KeyGenOpData 结构引用

CpaFlatBuffer _CpaCyDhPhase1KeyGenOpData::baseG

CpaFlatBuffe_CpaCyDhPhase1KeyGenOpData::base

Flat buffer containing a pointer to base (g). This MUST comply with the following: 0 < g < p.

包含指向 base (g)的指针的平面缓冲区。这必须符合以下条件:0 < g < p。

CpaFlatBuffer _CpaCyDhPhase1KeyGenOpData::privateValueX

Flat buffer containing a pointer to the private value (x). This is a random value which MUST satisfy the following condition: 0 < PrivateValueX < (PrimeP - 1)

CpaFlatBuffe_CpaCyDhPhase1KeyGenOpData::privateValue

Refer to PKCS #3: Diffie-Hellman Key-Agreement Standard for details. The client creating this data MUST ensure the compliance of this value with the standard. Note: This value is also needed to complete local phase 2 Diffie-Hellman operation.

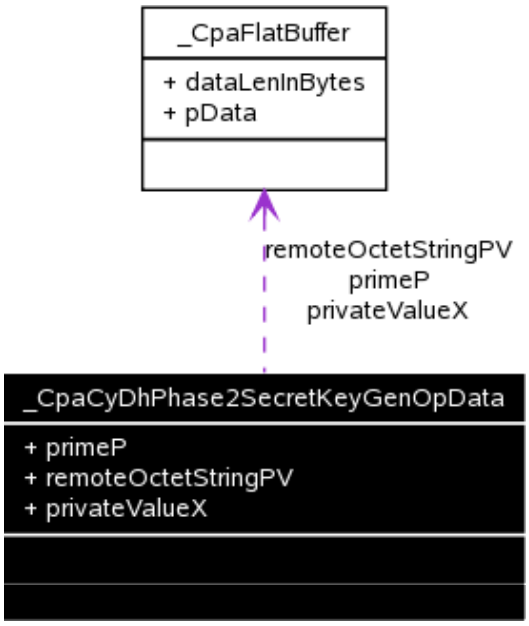
有关详细信息，请参考 PKCS #3: Diffie-Hellman 密钥协商标准。创建该数据的客户端必须确保该值符合标准。注意:完成本地阶段 2 Diffie-Hellman 操作也需要该值。

12.5.3 _CpaCyDhPhase2SecretKeyGenOpData Struct Reference

12.5.4 _ cpacydhphase 2 secretkeygenopdata 结构引用

Collaboration diagram for _CpaCyDhPhase2SecretKeyGenOpData:

_ cpacydhphase 2 secretkeygenopdata 的协作图:



12.5.4.1 Detailed Description

12.5.4.2 详细描述

Diffie-Hellman Phase 2 Secret Key Generation Data.
Diffie-Hellman 第 2 阶段密钥生成数据。

This structure lists the different items that required in the cpaCyDhKeyGenPhase2Secret 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 cpaCyDhKeyGenPhase2Secret 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 cpaCyDhKeyGenPhase2Secret function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order,
如果客户端在将此结构中引用的内存提交给 cpaCyDhKeyGenPhase2Secret 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。该结构中的所有值都要求以最高有效字节优先的顺序排列，
e.g. primeP.pData[0] = MSB.
例如 primeP.pData[0] = MSB。

12.5.4.3 Data Fields
12.5.4.4 数据字段

- **CpaFlatBuffer primeP**
- CpaFlatBuffer primeP
- **CpaFlatBuffer remoteOctetStringPV**
- CpaFlatBuffer remoteOctetStringPV

12.5.2 _CpaCyDhPhase2SecretKeyGenOpData Struct Reference

12.5.3_ cpacydhphase 2 secretkeygenopdata 结构引用

- **CpaFlatBuffer privateValueX**
- CpaFlatBuffer privateValueX

12.5.4.5 Field Documentation
12.5.4.6 现场文件

CpaFlatBuffer CpaCyDhPhase2SecretKeyGenOpData::primeP

Flat buffer containing a pointer to the random odd prime number (p). The bit-length of this number may be one of 768, 1024, 1536, 2048, 3072 or 4096. This SHOULD be same prime number as was used in the phase 1 key generation operation.
包含指向随机奇素数(p)的指针的平面缓冲区。这个数的位长可以是 768、1024、1536、2048、3072 或 4096 中的一个。这应该与阶段 1 密钥生成操作中使用的素数相同。

CpaFlatBuffer CpaCyDhPhase2SecretKeyGenOpData::remoteOctetStringPV

Flat buffer containing a pointer to the remote entity octet string Public Value (PV).
包含指向远程实体八位字节字符串公共值(PV)的指针的平面缓冲区。

CpaFlatBuffer CpaCyDhPhase2SecretKeyGenOpData::privateValueX

Flat buffer containing a pointer to the private value (x). This value may have been used in a call to the cpaCyDhKeyGenPhase1 function. This is a random value which MUST satisfy the following condition: 0 < privateValueX < (primeP - 1).
包含指向私有值(x)的指针的平面缓冲区。此值可能已在对 cpaCyDhKeyGenPhase1 函数的调用中使用。这是一个随机值，必须满足以下条件:0 < privateValueX < (primeP - 1)。

12.5.4 _CpaCyDhStats Struct Reference

12.5.5 _CpaCyDhStats 结构引用

12.5.3.1 Detailed Description

Diffie-Hellman Statistics.

Deprecated:

12.5.3.2 Diffie-Hellman 统

计。Deprecated:
As of v1.3 of the Crypto API, this structure has been deprecated, replaced by **CpaCyDhStats64**.
从 Crypto API 的 1.3 版开始，这种结构已被取代，由 CpaCyDhStats64

This structure contains statistics on the Diffie-Hellman operations. Statistics are set to zero when the component is initialized, and are collected per instance.
此结构包含 Diffie-Hellman 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。

12.5.3.3 Data Fields
12.5.3.4 数据字段

- **Cpa32U numDhPhase1KeyGenRequests**
- Cpa32U numDhPhase1KeyGenRequests
- **Cpa32U numDhPhase1KeyGenRequestErrors**
- Cpa32U numDhPhase1KeyGenRequestErrors
- **Cpa32U numDhPhase1KeyGenCompleted**
- Cpa32U numDhPhase1KeyGenCompleted
- **Cpa32U numDhPhase1KeyGenCompletedErrors**
- Cpa32U numDhPhase1KeyGenCompletedErrors
- **Cpa32U numDhPhase2KeyGenRequests**
- Cpa32U numDhPhase2KeyGenRequests
- **Cpa32U numDhPhase2KeyGenRequestErrors**
- Cpa32U numDhPhase2KeyGenRequestErrors
- **Cpa32U numDhPhase2KeyGenCompleted**
- Cpa32U numDhPhase2KeyGenCompleted
- **Cpa32U numDhPhase2KeyGenCompletedErrors**
- Cpa32U numDhPhase2KeyGenCompletedErrors

12.5.3.5 Field Documentation

12.5.3.6 现场文件

Cpa32U CpaCyDhStats::numDhPhase1KeyGenRequests

Total number of successful Diffie-Hellman phase 1 key generation requests.
成功的 Diffie-Hellman 阶段 1 密钥生成请求的总数。

Cpa32U CpaCyDhStats::numDhPhase1KeyGenRequestErrors

Total number of Diffie-Hellman phase 1 key generation requests that had an error and could not be processed.
出现错误且无法处理的 Diffie-Hellman 阶段 1 密钥生成请求的总数。

Cpa32U CpaCyDhStats::numDhPhase1KeyGenCompleted

Total number of Diffie-Hellman phase 1 key generation operations that completed successfully.
成功完成的 Diffie-Hellman 阶段 1 密钥生成操作的总数。

12.5.3 _CpaCyDhStats Struct Reference

12.5.4_CpaCyDhStats 结构引用

Total number of Diffie-Hellman phase 1 key generation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 Diffie-Hellman 阶段 1 密钥生成操作的总数。

Total number of successful Diffie-Hellman phase 2 key generation requests.
成功的 Diffie-Hellman 阶段 2 密钥生成请求的总数。

Total number of Diffie-Hellman phase 2 key generation requests that had an error and could not be processed.
出现错误且无法处理的 Diffie-Hellman 阶段 2 密钥生成请求的总数。

Total number of Diffie-Hellman phase 2 key generation operations that completed successfully.
成功完成的 Diffie-Hellman 阶段 2 密钥生成操作的总数。

Total number of Diffie-Hellman phase 2 key generation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 Diffie-Hellman 阶段 2 密钥生成操作的总数。

12.5.5 _CpaCyDhStats64 Struct Reference

12.5.6 _CpaCyDhStats64 结构引用

12.5.6.1 Detailed Description
12.5.6.2 详细描述

Diffie-Hellman Statistics (64-bit version).
Diffie-Hellman 统计信息(64 位版本)。

This structure contains the 64-bit version of the statistics on the Diffie-Hellman operations. Statistics are set to zero when the component is initialized, and are collected per instance.
此结构包含 64 位版本的 Diffie-Hellman 操作统计信息。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。

12.5.6.3 Data Fields
12.5.6.4 数据字段

- **Cpa64U numDhPhase1KeyGenRequests**
- Cpa64U numDhPhase1KeyGenRequests
- **Cpa64U numDhPhase1KeyGenRequestErrors**
- Cpa64U numDhPhase1KeyGenRequestErrors
- **Cpa64U numDhPhase1KeyGenCompleted**
- Cpa64U numDhPhase1KeyGenCompleted
- **Cpa64U numDhPhase1KeyGenCompletedErrors**
- Cpa64U numDhPhase1KeyGenCompletedErrors

- **Cpa64U numDhPhase2KeyGenRequests**
- Cpa64U numDhPhase2KeyGenRequests
- **Cpa64U numDhPhase2KeyGenRequestErrors**
- Cpa64U numDhPhase2KeyGenRequestErrors
- **Cpa64U numDhPhase2KeyGenCompleted**
- Cpa64U numDhPhase2KeyGenCompleted
- **Cpa64U numDhPhase2KeyGenCompletedErrors**
- Cpa64U numDhPhase2KeyGenCompletedErrors

12.5.6.5 Field Documentation
12.5.6.6 现场文件

Total number of successful Diffie-Hellman phase 1 key generation requests.
成功的 Diffie-Hellman 阶段 1 密钥生成请求的总数。

Total number of Diffie-Hellman phase 1 key generation requests that had an error and could not be processed.
出现错误且无法处理的 Diffie-Hellman 阶段 1 密钥生成请求的总数。

Total number of Diffie-Hellman phase 1 key generation operations that completed successfully.
成功完成的 Diffie-Hellman 阶段 1 密钥生成操作的总数。

Total number of Diffie-Hellman phase 1 key generation operations that could not be completed successfully due to errors.

由于错误而无法成功完成的 Diffie-Hellman 阶段 1 密钥生成操作的总数。

Total number of successful Diffie-Hellman phase 2 key generation requests.
成功的 Diffie-Hellman 阶段 2 密钥生成请求的总数。

Total number of Diffie-Hellman phase 2 key generation requests that had an error and could not be processed.
出现错误且无法处理的 Diffie-Hellman 阶段 2 密钥生成请求的总数。

Total number of Diffie-Hellman phase 2 key generation operations that completed successfully.
成功完成的 Diffie-Hellman 阶段 2 密钥生成操作的总数。

Total number of Diffie-Hellman phase 2 key generation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 Diffie-Hellman 阶段 2 密钥生成操作的总数。

12.7 Typedef Documentation

12.8 Typedef 文档

Diffie-Hellman Phase 1 Key Generation Data.
Diffie-Hellman 第一阶段密钥生成数据。

This structure lists the different items that are required in the cpaCyDhKeyGenPhase1 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 CpaCyDhPhase1KeyGenOpData structure.
此结构列出了 cpaCyDhKeyGenPhase1 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。当这个结构与 CpaCyDhPhase1KeyGenOpData 结构一起返回时，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 cpaCyDhKeyGenPhase1 function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order,
如果客户端在将此结构中引用的内存提交给 cpaCyDhKeyGenPhase1 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。该结构中的所有值都要求以最高有效字节优先的顺序排列，
e.g. primeP.pData[0] = MSB.
例如 primeP.pData[0] = MSB。

Diffie-Hellman Phase 2 Secret Key Generation Data.

Diffie-Hellman 第 2 阶段密钥生成数据。

This structure lists the different items that required in the cpaCyDhKeyGenPhase2Secret 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 cpaCyDhKeyGenPhase2Secret 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 cpaCyDhKeyGenPhase2Secret function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. primeP.pData[0] = MSB.
如果客户端在将此结构中引用的内存提交给 cpaCyDhKeyGenPhase2Secret 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。该结构中的所有值都要求以最高有效字节优先，例如 primeP.pData[0] = MSB。

typedef struct **CpaCyDhStats CPA_DEPRECATED**

Diffie-Hellman Statistics.
迪菲-赫尔曼统计学。

Deprecated:
Deprecated:

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by **CpaCyDhStats64**.
从 Crypto API 的 1.3 版开始，这种结构已被取代，由 CpaCyDhStats64

12.6 Typedef Documentation

12.7 Typedef 文档

This structure contains statistics on the Diffie-Hellman operations. Statistics are set to zero when the component is initialized, and are collected per instance.

此结构包含 Diffie-Hellman 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 _**CpaCyDhStats64 CpaCyDhStats64**

typedef 结构_CpaCyDhStats6CpaCyDhStats6

Diffie-Hellman Statistics (64-bit version).

Diffie-Hellman 统计信息(64 位版本)。

This structure contains the 64-bit version of the statistics on the Diffie-Hellman operations. Statistics are set to zero when the component is initialized, and are collected per instance.

此结构包含 64 位版本的 Diffie-Hellman 操作统计信息。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。

12.8 Function Documentation

12.9 功能文档

```
cpaCyDhKeyGenPhase1 (  const                                    instanceHandle, pDhPhase1Cb,
cpaCyDhKeyGenPhase1 (  const                                    instanceHandle, pDhPhase1Cb,
const
                                  void *pCallbackTag,
                                  const *  pPhase1KeyGenData,
                                  *pLocalOctetStringPV
                                )
```

Function to implement Diffie-Hellman phase 1 operations.

函数来实现 Diffie-Hellman 阶段 1 操作。

This function may be used to implement the Diffie-Hellman phase 1 operations as defined in the PKCS #3 standard. It may be used to generate the the (local) octet string public value (PV) key. The prime number sizes specified in RFC 2409, 4306, and part of RFC 3526 are supported (bit sizes 6144 and 8192 from RFC 3536 are not supported).

该函数可用于实现 PKCS #3 标准中定义的 Diffie-Hellman 阶段 1 操作。它可用于生成(本地)八位字节字符串公共值(PV)密钥。支持 RFC 2409、4306 和部分 RFC 3526 中指定的素数大小(不支持 RFC 3536 中的位大小 6144 和 8192)。

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] *pDhPhase1Cb* Pointer to a callback function to be invoked when the operation is complete. If the pointer is set to a NULL value the function will operate synchronously.
[in] pDhPhase1Cb 指标，指向当作业为时要叫用的回呼函式
完成。如果指针设置为空值，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。将原封不动地退回
在回调中

[in] *pPhase1KeyGenData* Structure containing all the data needed to perform the DH Phase 1 key generation operation. The client code allocates the memory for
[in] pPhase1KeyGenData 结构，包含执行 DH 阶段 1 所需的所有数据
密钥生成操作。客户端代码为分配内存

this structure. This component takes ownership of the memory until it is returned in the callback.

这个结构。该组件取得内存的所有权，直到它在回调中被返回。

[out] *pLocalOctetStringPV* Pointer to memory allocated by the client into which the (local) octet

[out]指向客户端分配的内存的 pLocalOctetStringPV 指针

string Public Value (PV) will be written. This value needs to be sent to the remote entity with which Diffie-Hellman is negotiating. The size of this buffer in bytes (as represented by the dataLenInBytes field) MUST be at least big enough to store the public value, which may have a bit length up to that of pPrimeP. On invocation the callback function will contain this parameter in the pOut parameter.

将写入字符串公共值(PV)。这个值需要发送到 Diffie-Hellman 与之协商的远程实体。该缓冲区的字节大小(由 dataLenInBytes 字段表示)必须至少足够大，以存储公共值，公共值的位长可以达到 pPrimeP 的位长。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_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 pDhPhase1Cb is non-NULL an asynchronous callback of type CpaCyGenFlatBufCbFunc is generated in response to this function call. Any errors generated during processing are reported in the structure returned in the callback.
当 pDhPhase1Cb 为非 NULL 时，将生成一个 CpaCyGenFlatBufCbFunc 类型的异步回调来响应此函数调用。处理过程中生成的任何错误都在回调中返回的结构中报告。

See also:

另请参见：

CpaCyGenFlatBufCbFunc, **CpaCyDhPhase1KeyGenOpData**
CpaCyGenFlatBufCbFunc, CpaCyDhPhase1KeyGenOpData

```
cpaCyDhKeyGenPhase2Secret( const                                    instanceHandle,
const                                                               pDhPhase2Cb, pCallbackTag,
void *
                                const
                                  pPhase2SecretKeyGenData,
                                *
                                *pOctetStringSecretKey
                               )
```

Function to implement Diffie-Hellman phase 2 operations.
函数来实现 Diffie-Hellman 阶段 2 操作。

This function may be used to implement the Diffie-Hellman phase 2 operation as defined in the PKCS #3 standard. It may be used to generate the Diffie-Hellman shared secret key.
该函数可用于实现 PKCS #3 标准中定义的 Diffie-Hellman 阶段 2 操作。它可用于生成 Diffie-Hellman 共享密钥。

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]	*pDhPhase2Cb*	Pointer to a callback function to be invoked when the operation is complete. If the pointer is set to a NULL value the function will operate synchronously. [in] pDhPhase2Cb 指标，指向当操作完成。如果指针设置为空值，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。将被归还回调中不变。
[in]	*pPhase2SecretKeyGenData*	Structure containing all the data needed to perform the DH Phase 2 secret key generation 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] pPhase2SecretKeyGenData 结构包含执行 DH 所需的所有数据第二阶段密钥生成操作。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]	*pOctetStringSecretKey*	Pointer to memory allocated by the client into which the octet string secret key will be written. The size of this buffer in bytes (as represented by the dataLenInBytes field) MUST be at least big enough to store the public value, which may have a bit length up to that of pPrimeP. On invocation the callback function will contain this parameter in the pOut parameter. [out] pOctetStringSecretKey 指向由客户端分配的内存的指针将写入字符串密钥。该缓冲区的字节大小(由 dataLenInBytes 字段表示)必须至少足够大，以存储公共值，公共值的位长可以达到 pPrimeP 的位长。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_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 pDhPha	se2Cb is non-NULL an asynchronous callback of type CpaCyGenFlatBufCbFunc is generated in response to this function call. Any errors generated during processing are reported in the structure returned in the callback. 当 pDhPhase2Cb 为非 NULL 时，将生成一个 CpaCyGenFlatBufCbFunc 类型的异步回调来响应此函数调用。处理过程中生成的任何错误都在回调中返回的结构中报告。

See also:

另请参见：

CpaCyGenFlatBufCbFunc, **CpaCyDhPhase2SecretKeyGenOpData**
CpaCyGenFlatBufCbFunc, CpaCyDhPhase2SecretKeyGenOpData

CpaStatus CPA_DEPRECATED cpaCyDhQueryStats (const **CpaInstanceHandle** *instanceHandle*,
struct **CpaCyDhStats** * *pDhStats*

CpaStatus CPA_DEPRECATED cpaCyDhQueryStats (const CpaInstanceHandle *instanceHandle*,
struct _CpaCyDhStats * *pDhStats*
)

Query statistics for Diffie-Hellman operations
Diffie-Hellman 操作的查询统计信息

Deprecated:
Deprecated:
As of v1.3 of the Crypto API, this function has been deprecated, replaced by
从 Crypto API 1.3 版开始，此函数已被弃用，由
cpaCyDhQueryStats64().
cpaCyDhQueryStats64()。

This function will query a specific Instance handle for Diffie- Hellman statistics. The user MUST allocate the CpaCyDhStats structure and pass the reference to that structure into this function call. This function writes the statistic results into the passed in CpaCyDhStats structure.
这个函数将查询 Diffie- Hellman 统计数据的特定实例句柄。用户必须分配 CpaCyDhStats 结构，并将对该结构的引用传递到这个函数调用中。该函数将统计结果写入传入的 CpaCyDhStats 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

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] *pDhStats* Pointer to memory into which the statistics will be written.
[out] pDhStats 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_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。
CPA_STATUS_UNSUPPORTED Function is not supported.
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:
CpaCyDhStats
CpaCyDhStats

CpaStatus cpaCyDhQueryStats64 (const **CpaInstanceHandle** *instanceHandle*,
CpaCyDhStats64 * *pDhStats*
)

CpaStatus cpaCyDhQueryStats64 (const CpaInstanceHandle *instanceHandle*,
CpaCyDhStats64 * *pDhStats*
)

Query statistics (64-bit version) for Diffie-Hellman operations
Diffie-Hellman 操作的查询统计信息(64 位版本)

This function will query a specific Instance handle for the 64-bit version of the Diffie-Hellman statistics. The user MUST allocate the CpaCyDhStats64 structure and pass the reference to that structure into this function call. This function writes the statistic results into the passed in CpaCyDhStats64 structure.
此函数将查询 64 位版本的 Diffie-Hellman 统计信息的特定实例句柄。用户必须分配 CpaCyDhStats64 结构，并将对该结构的引用传递到此函数调用中。此函数将统计结果写入传入的 CpaCyDhStats64 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

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] *pDhStats* Pointer to memory into which the statistics will be written.
[out] pDhStats 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_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。
CPA_STATUS_UNSUPPORTED Function is not supported.
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:

CpaCyDhStats64
CpaCyDhStats64

16 Digital Signature Algorithm (DSA) API

17 数字签名算法(DSA) API

[Cryptographic API]

[Cryptographic API]

Collaboration diagram for Digital Signature Algorithm (DSA) API:
数字签名算法(DSA) API 的协作图:

17.1 Detailed Description

17.2 详细描述

File: cpa_cy_dsa.h

文件:cpa_cy_dsa.h

These functions specify the API for Public Key Encryption (Cryptography) Digital Signature Algorithm (DSA) operations.
这些函数指定了用于公钥加密(加密)数字签名算法(DSA)操作的 API。

Support is provided for FIPS PUB 186-2 with Change Notice 1 specification, and optionally for FIPS PUB 186-3. If an implementation does not support FIPS PUB 186-3, then the corresponding functions may return a status of **CPA_STATUS_FAIL**.
支持 FIPS 出版公司 186-2 的变更通告 1 规范，并可选择支持 FIPS 出版公司 186-3. 如果实现不支持 FIPS 发布 186-3，则相应的函数可以返回状态 CPA_STATUS_FAIL

Support for FIPS PUB 186-2 with Change Notice 1 implies supporting the following choice for the pair L and N:
支持具有变更通知 1 的 FIPS 公共 186-2 意味着支持对 L 和 N 的以下选择:

- L = 1024, N = 160
- L = 1024，N = 160

Support for FIPS PUB 186-3 implies supporting the following choices for the pair L and N:
支持 FIPS 公共 186-3 意味着支持对 L 和 N 的以下选择:

- L = 1024, N = 160
- L = 1024，N = 160
- L = 2048, N = 224
- L = 2048，N = 224
- L = 2048, N = 256
- L = 2048，N = 256
- L = 3072, N = 256
- L = 3072，N = 256

Only the modular math aspects of DSA parameter generation and message signature generation and verification are implemented here. For full DSA support, this DSA API SHOULD be used in conjunction with other parts of this overall Cryptographic API. In particular the Symmetric functions (for hashing), the Random Number Generation functions, and the Prime Number Test functions will be required.
这里只实现了 DSA 参数生成以及消息签名生成和验证的模块化数学方面。为了获得完整的 DSA 支持，这个 DSA API 应该与这个整体加密 API 的其他部分结合使用。特别是对称函数(用于散列)、随机数生成函数和素数测试函数将是必需的。

Note:
注意：

Large numbers are represented on the QuickAssist API as described in the Large Number API (**Cryptographic Large Number API**).
大数在 QuickAssist API 上表示，如大数 API(Cryptographic Large Number API

17.3 Data Structures
17.4 数据结构

- struct _**CpaCyDsaPParamGenOpData**

- 结构体_CpaCyDsaPParamGenOpData
- struct _**CpaCyDsaGParamGenOpData**
- 结构体_CpaCyDsaGParamGenOpData
- struct _**CpaCyDsaYParamGenOpData**
- 结构体_CpaCyDsaYParamGenOpData
- struct _**CpaCyDsaRSignOpData**
- 结构体_CpaCyDsaRSignOpData
- struct _**CpaCyDsaSSignOpData**
- 结构体_CpaCyDsaSSignOpData
- struct _**CpaCyDsaRSSignOpData**
- 结构体_CpaCyDsaRSSignOpData
- struct _**CpaCyDsaVerifyOpData**
- 结构体_CpaCyDsaVerifyOpData
- struct _**CpaCyDsaStats**
- 结构体_CpaCyDsaStats
- struct _**CpaCyDsaStats64**
- 结构体_CpaCyDsaStats64

17.5 Typedefs
17.6 类型定义

13.3 Typedefs

13.4 类型定义

- typedef _**CpaCyDsaPParamGenOpData CpaCyDsaPParamGenOpData**

- 数据类型说明_CpaCyDsaPParamGenOpData CpaCyDsaPParamGenOpData
- typedef _**CpaCyDsaGParamGenOpData CpaCyDsaGParamGenOpData**
- 数据类型说明_CpaCyDsaGParamGenOpData CpaCyDsaGParamGenOpData
- typedef _**CpaCyDsaYParamGenOpData CpaCyDsaYParamGenOpData**
- 数据类型说明_CpaCyDsaYParamGenOpData CpaCyDsaYParamGenOpData
- typedef _**CpaCyDsaRSignOpData CpaCyDsaRSignOpData**
- 数据类型说明_CpaCyDsaRSignOpData CpaCyDsaRSignOpData
- typedef _**CpaCyDsaSSignOpData CpaCyDsaSSignOpData**
- 数据类型说明_CpaCyDsaSSignOpData CpaCyDsaSSignOpData
- typedef _**CpaCyDsaRSSignOpData CpaCyDsaRSSignOpData**
- 数据类型说明_CpaCyDsaRSSignOpData CpaCyDsaRSSignOpData
- typedef _**CpaCyDsaVerifyOpData CpaCyDsaVerifyOpData**
- 数据类型说明_CpaCyDsaVerifyOpData CpaCyDsaVerifyOpData
- typedef _**CpaCyDsaStats CPA_DEPRECATED**
- 数据类型说明_CpaCyDsaStats CPA_DEPRECATED
- typedef _**CpaCyDsaStats64 CpaCyDsaStats64**
- 数据类型说明_CpaCyDsaStats64 CpaCyDsaStats64
- typedef void(* **CpaCyDsaGenCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpData,
- typedef void(*CpaCyDsaGenCbFunc CpaStatus
  CpaBoolean protocolStatus, **CpaFlatBuffer** *pOut)
  CpaBoolean 协议状态，CpaFlatBuffer
- typedef void(* **CpaCyDsaRSSignCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpData,
- typedef void(*CpaCyDsaRSSignCbFunc CpaStatus
  CpaBoolean protocolStatus, **CpaFlatBuffer** *pR, **CpaFlatBuffer** *pS)
  CpaBoolean 协议状态，CpaFlatBuffer CpaFlatBuffer
- typedef void(* **CpaCyDsaVerifyCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpData,
- typedef void(*CpaCyDsaVerifyCbFunc CpaStatus
  CpaBoolean verifyStatus)
  CpaBoolean 验证状态）

13.5 Functions
13.6 功能

- **CpaStatus cpaCyDsaGenPParam** (const **CpaInstanceHandle** instanceHandle, const **CpaCyDsaGenCbFunc** pCb, void *pCallbackTag, const **CpaCyDsaPParamGenOpData** *pOpData, **CpaBoolean** *pProtocolStatus, **CpaFlatBuffer** *pP)

- CpaStatus cpaCyDsaGenPParam （常量CpaInstanceHandle CpaCyDsaGenCbFunc
  CpaCyDsaPParamGenOpData CpaBoolean CpaFlatBuffer
- **CpaStatus cpaCyDsaGenGParam** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyDsaGenGParam （常量CpaInstanceHandle
  CpaCyDsaGenCbFunc pCb, void *pCallbackTag, const **CpaCyDsaGParamGenOpData** *pOpData,
  CpaCyDsaGenCbFunc pCb，void *pCallbackTag，constCpaCyDsaGParamGenOpData
  CpaBoolean *pProtocolStatus, **CpaFlatBuffer** *pG)
  CpaBoolean *协议状态，CpaFlatBuffer
- **CpaStatus cpaCyDsaGenYParam** (const **CpaInstanceHandle** instanceHandle, const **CpaCyDsaGenCbFunc** pCb, void *pCallbackTag, const **CpaCyDsaYParamGenOpData** *pOpData,

CpaBoolean *pProtocolStatus, **CpaFlatBuffer** *pY)
- CpaStatus cpaCyDsaGenYParam（常量 CpaInstanceHandle CpaCyDsaGenCbFunc CpaCyDsaYParamGenOpData CpaBoolean CpaFlatBuffer
- **CpaStatus cpaCyDsaSignR** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyDsaSignR（常量 CpaInstanceHandle **CpaCyDsaGenCbFunc** pCb, void *pCallbackTag, const **CpaCyDsaRSignOpData** *pOpData, CpaCyDsaGenCbFunc pCb，void *pCallbackTag，constCpaCyDsaRSignOpData **CpaBoolean** *pProtocolStatus, **CpaFlatBuffer** *pR) CpaBoolean *协议状态，CpaFlatBuffer
- **CpaStatus cpaCyDsaSignS** (const **CpaInstanceHandle** instanceHandle, const **CpaCyDsaGenCbFunc** pCb, void *pCallbackTag, const **CpaCyDsaSSignOpData** *pOpData, **CpaBoolean** *pProtocolStatus, **CpaFlatBuffer** *pS)
- CpaStatus cpaCyDsaSignS（常量 CpaInstanceHandle CpaCyDsaGenCbFunc CpaCyDsaSSignOpData CpaBoolean CpaFlatBuffer
- **CpaStatus cpaCyDsaSignRS** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyDsaSignRS（常量 CpaInstanceHandle **CpaCyDsaRSSignCbFunc** pCb, void *pCallbackTag, const **CpaCyDsaRSSignOpData** *pOpData, CpaCyDsaRSSignCbFunc pCb，void *pCallbackTag，constCpaCyDsaRSSignOpData **CpaBoolean** *pProtocolStatus, **CpaFlatBuffer** *pR, **CpaFlatBuffer** *pS) CpaBoolean *协议状态，CpaFlatBuffer CpaFlatBuffer
- **CpaStatus cpaCyDsaVerify** (const **CpaInstanceHandle** instanceHandle, const **CpaCyDsaVerifyCbFunc** pCb, void *pCallbackTag, const **CpaCyDsaVerifyOpData** *pOpData, **CpaBoolean** *pVerifyStatus)
- CpaStatus cpaCyDsaVerify（常量 CpaInstanceHandle CpaCyDsaVerifyCbFunc CpaCyDsaVerifyOpData CpaBoolean
- **CpaStatus CPA_DEPRECATED cpaCyDsaQueryStats** (const **CpaInstanceHandle**
- CpaStatus CPA_DEPRECATED cpaCyDsaQueryStats（常量 CpaInstanceHandle instanceHandle, struct **_CpaCyDsaStats** *pDsaStats) instanceHandle，结构_CpaCyDsaStats
- **CpaStatus cpaCyDsaQueryStats64** (const **CpaInstanceHandle** instanceHandle,
- CpaStatus cpaCyDsaQueryStats64（常量 CpaInstanceHandle **CpaCyDsaStats64** *pDsaStats) CpaCyDsaStats64 * pDsaStats)

13.7 Data Structure Documentation

13.8 数据结构文档

13.5 Data Structure Documentation

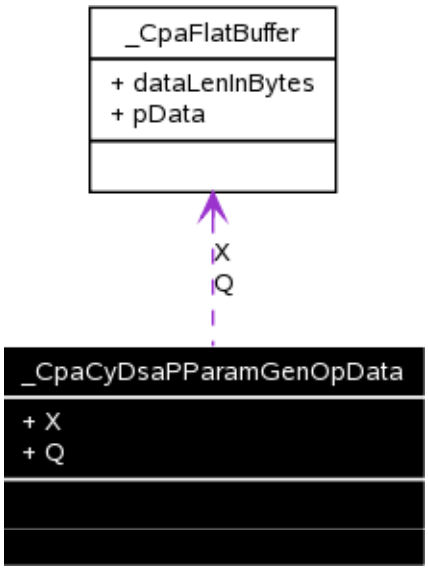
13.6 数据结构文档

13.6.1 _CpaCyDsaPParamGenOpData Struct Reference

13.6.2 _ CpaCyDsaPParamGenOpData 结构引用

Collaboration diagram for _CpaCyDsaPParamGenOpData:

_ CpaCyDsaPParamGenOpData 的协作图:



13.6.2.1 Detailed Description
13.6.2.2 详细描述

DSA P Parameter Generation Operation Data.
DSA P 参数生成操作数据。

This structure contains the operation data for the cpaCyDsaGenPParam 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 cpaCyDsaGenPParam 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. X.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 X.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 cpaCyDsaGenPParam 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 cpaCyDsaGenPParam 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：

cpaCyDsaGenPParam()
cpaCyDsaGenPParam()

13.6.2.3 Data Fields
13.6.2.4 数据字段

- **CpaFlatBuffer X**
- CpaFlatBuffer X
- **CpaFlatBuffer Q**
- CpaFlatBuffer Q

13.6.2.5 Field Documentation
13.6.2.6 现场文件

CpaFlatBuffer _CpaCyDsaPParamGenOpData::X
CpaFlatBuffe_CpaCyDsaPParamGenOpData::
2^(L-1) <= X < 2^L (from FIPS 186-3)
2^(L-1) <= X < 2^L(来自 FIPS 186-3)

13.5.1 _CpaCyDsaPParamGenOpData Struct Reference

13.5.2_ CpaCyDsaPParamGenOpData 结构引用

CpaFlatBuffer _CpaCyDsaPParamGenOpData::Q

CpaFlatBuffe_CpaCyDsaPParamGenOpData::

DSA group parameter q

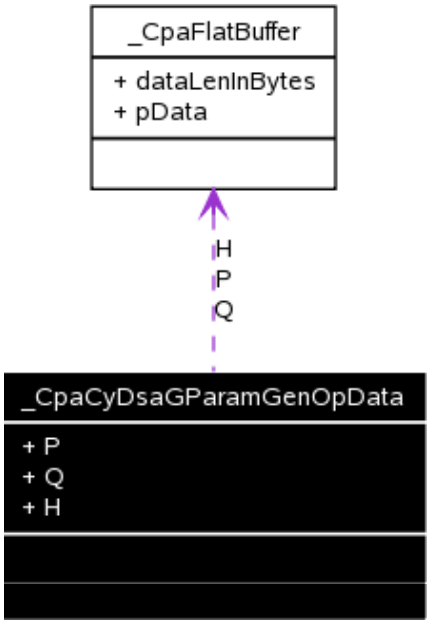
DSA 组参数 q

13.5.3 _CpaCyDsaGParamGenOpData Struct Reference

13.5.4 _CpaCyDsaGParamGenOpData 结构引用

Collaboration diagram for _CpaCyDsaGParamGenOpData:

_CpaCyDsaGParamGenOpData 的协作图:



13.5.4.1 Detailed Description
13.5.4.2 详细描述

DSA G Parameter Generation Operation Data.
DSA G 参数生成操作数据。

This structure contains the operation data for the cpaCyDsaGenGParam 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 cpaCyDsaGenGParam 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。

All values in this structure are required to be in Most Significant Byte first order, e.g. P.pData[0] = MSB.

All numbers MUST be stored in big-endian order.

该结构中的所有值都要求以最高有效字节优先，例如 P.pData[0] = MSB。所有数字都必须以大端顺序存储。

Note:
注意：

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDsaGenGParam 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 cpaCyDsaGenGParam 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：

cpaCyDsaGenGParam()
cpaCyDsaGenGParam()

13.5.4.3 Data Fields
13.5.4.4 数据字段

- **CpaFlatBuffer P**
- CpaFlatBuffer P
- **CpaFlatBuffer Q**
- CpaFlatBuffer Q
- **CpaFlatBuffer H**
- CpaFlatBuffer H

13.5.2 _CpaCyDsaGParamGenOpData Struct Reference

13.5.3_CpaCyDsaGParamGenOpData 结构引用

13.5.4.5 Field Documentation

13.5.4.6 现场文件

DSA group parameter p

DSA 组参数 p

DSA group parameter q

DSA 组参数 q

any integer with 1 < h < p - 1

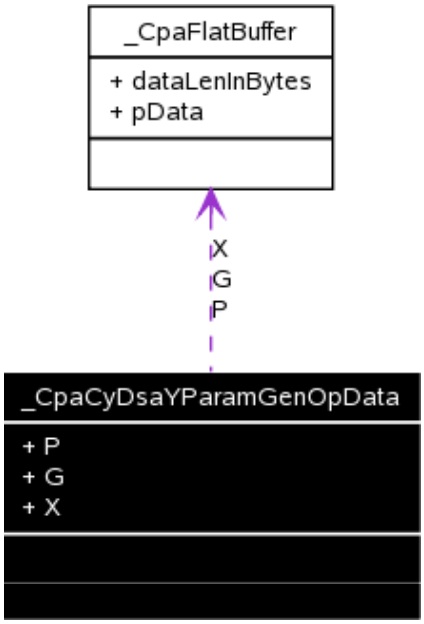
1 < h < p - 1 的任何整数

13.5.4 _CpaCyDsaYParamGenOpData Struct Reference

13.5.5 _CpaCyDsaYParamGenOpData 结构引用

Collaboration diagram for _CpaCyDsaYParamGenOpData:

_CpaCyDsaYParamGenOpData 的协作图:



13.5.5.1 Detailed Description

13.5.5.2 详细描述

DSA Y Parameter Generation Operation Data.
DSA Y 参数生成操作数据。

This structure contains the operation data for the cpaCyDsaGenYParam 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 cpaCyDsaGenYParam 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 SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. P.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 P.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 cpaCyDsaGenYParam 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 cpaCyDsaGenYParam 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。

13.5.3 _CpaCyDsaYParamGenOpData Struct Reference

13.5.4 _CpaCyDsaYParamGenOpData 结构引用

See also:

另请参见:

cpaCyDsaGenYParam()
cpaCyDsaGenYParam()

13.5.5.3 Data Fields
13.5.5.4 数据字段

- **CpaFlatBuffer P**
- CpaFlatBuffer P
- **CpaFlatBuffer G**
- CpaFlatBuffer G
- **CpaFlatBuffer X**
- CpaFlatBuffer X

13.5.5.5 Field Documentation
13.5.5.6 现场文件

DSA group parameter p
DSA 组参数 p

DSA group parameter g
DSA 组参数 g

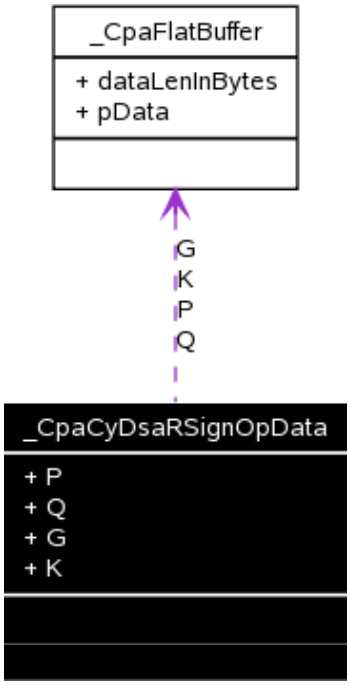
DSA private key x
DSA 私钥 x

13.5.5 _CpaCyDsaRSignOpData Struct Reference

13.5.6 _CpaCyDsaRSignOpData 结构引用

Collaboration diagram for _CpaCyDsaRSignOpData:

_CpaCyDsaRSignOpData 的协作图:



13.5.6.1 Detailed Description
13.5.6.2 详细描述

DSA R Sign Operation Data.
DSA R 标志操作数据。

This structure contains the operation data for the cpaCyDsaSignR 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
该结构包含了 cpaCyDsaSignR 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。内存的所有权返回给客户端

13.5.4 _CpaCyDsaRSignOpData Struct Reference

when this structure is returned in the callback function.

13.5.5在回调函数中返回此结构时的

_CpaCyDsaRSignOpData 结构引用。

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

All values in this structure are required to be in Most Significant Byte first order, e.g. P.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 P.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 cpaCyDsaSignR 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 cpaCyDsaSignR 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：

cpaCyDsaSignR()
cpaCyDsaSignR()

13.5.6.3 Data Fields
13.5.6.4 数据字段

- **CpaFlatBuffer P**
- CpaFlatBuffer P
- **CpaFlatBuffer Q**
- CpaFlatBuffer Q
- **CpaFlatBuffer G**
- CpaFlatBuffer G
- **CpaFlatBuffer K**
- CpaFlatBuffer K

13.5.6.5 Field Documentation
13.5.6.6 现场文件

DSA group parameter p
DSA 组参数 p

DSA group parameter q
DSA 组参数 q

DSA group parameter g

DSA 组参数 g

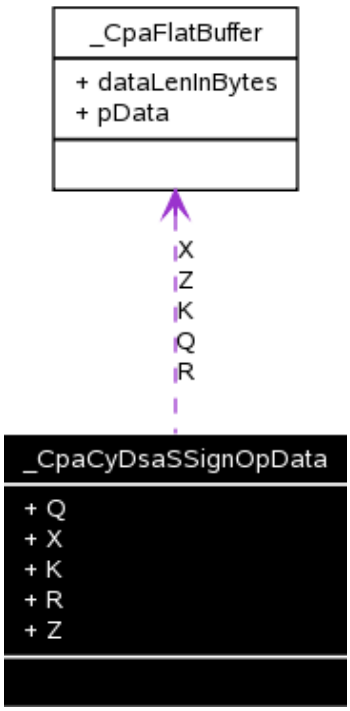
DSA secret parameter k for signing
用于签名的 DSA 秘密参数 k

13.5.6 _CpaCyDsaSSignOpData Struct Reference

13.5.7 _ CpaCyDsaSSignOpData 结构引用

Collaboration diagram for _CpaCyDsaSSignOpData:

_CpaCyDsaSSignOpData 的协作图:



13.5.7.1 Detailed Description

13.5.7.2 详细描述

DSA S Sign Operation Data.
DSA 的签名操作数据。

This structure contains the operation data for the cpaCyDsaSignS 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 cpaCyDsaSignS 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 SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. Q.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 Q.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

cpaCyDsaSignS 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 cpaCyDsaSignS 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：

cpaCyDsaSignS()

cpaCyDsaSignS()

13.5.7.3 Data Fields

13.5.7.4 数据字段

- **CpaFlatBuffer Q**
- CpaFlatBuffer Q
- **CpaFlatBuffer X**
- CpaFlatBuffer X
- **CpaFlatBuffer K**
- CpaFlatBuffer K
- **CpaFlatBuffer R**
- CpaFlatBuffer R
- **CpaFlatBuffer Z**
- CpaFlatBuffer Z

13.5.5 CpaCyDsaRSSignOpData Struct

13.5.7.5 Field Documentation

13.5.7.6 现场文件

CpaFlatBuffer CpaCyDsaRSSignOpData::q

DSA group parameter q

DSA 组参数 q

CpaFlatBuffer CpaCyDsaRSSignOpData::x

DSA private key x

DSA 私钥 x

CpaFlatBuffer CpaCyDsaRSSignOpData::k

DSA secret parameter k for signing

用于签名的 DSA 秘密参数 k

CpaFlatBuffer CpaCyDsaRSSignOpData::r

DSA message signature r

DSA 消息签名 r

CpaFlatBuffer CpaCyDsaRSSignOpData::z

The leftmost min(N, outlen) bits of Hash(M), where:

Hash(M)最左边的 min(N, outlen)位，其中:

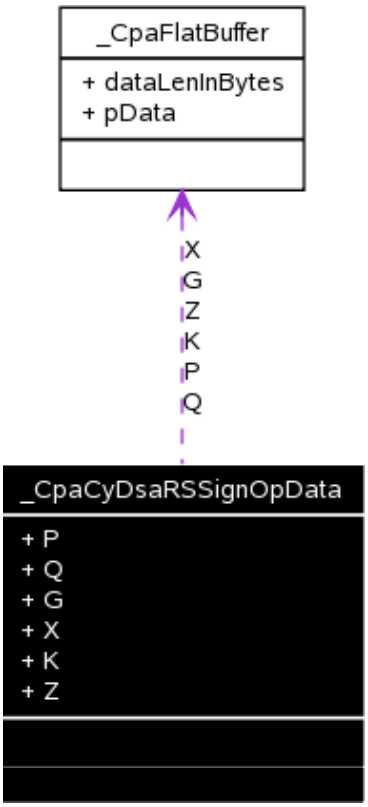
- N is the bit length of q
- n 是 q 的比特长度
- outlen is the bit length of the hash function output block
- outlen 是哈希函数输出模块的位长
- M is the message to be signed
- m 是要签名的消息

13.5.8 _CpaCyDsaRSSignOpData Struct Reference

13.5.9 _CpaCyDsaRSSignOpData 结构引用

Collaboration diagram for _CpaCyDsaRSSignOpData:

_CpaCyDsaRSSignOpData 的协作图:


_CpaFlatBuffer
+ dataLenInBytes
+ pData
X
G
Z
K
P
Q
_CpaCyDsaRSSignOpData
+ P
+ Q
+ G
+ X
+ K
+ Z

13.5.6 _CpaCyDsaRSSignOpData Struct Reference

13.5.7_CpaCyDsaRSSignOpData 结构引用

13.5.7.1 Detailed Description

13.5.7.2 详细描述

DSA R & S Sign Operation Data.
DSA R & S 标志操作数据。

This structure contains the operation data for the cpaCyDsaSignRS 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.
该结构包含了 cpaCyDsaSignRS 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 SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. P.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 P.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 cpaCyDsaSignRS 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 cpaCyDsaSignRS 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:

cpaCyDsaSignRS()
cpaCyDsaSignRS()

13.5.7.3 Data Fields
13.5.7.4 数据字段

- **CpaFlatBuffer P**
- CpaFlatBuffer P
- **CpaFlatBuffer Q**
- CpaFlatBuffer Q
- **CpaFlatBuffer G**
- CpaFlatBuffer G
- **CpaFlatBuffer X**
- CpaFlatBuffer X
- **CpaFlatBuffer K**
- CpaFlatBuffer K
- **CpaFlatBuffer Z**

- CpaFlatBuffer Z

13.5.7.5 Field Documentation
13.5.7.6 现场文件

DSA group parameter p
DSA 组参数 p

DSA group parameter q
DSA 组参数 q

DSA group parameter g
DSA 组参数 g

DSA private key x
DSA 私钥 x

DSA secret parameter k for signing
用于签名的 DSA 秘密参数 k

The leftmost min(N, outlen) bits of Hash(M), where:
Hash(M)最左边的 min(N，outlen)位，其中：

- N is the bit length of q
- n 是 q 的比特长度
- outlen is the bit length of the hash function output block
- outlen 是哈希函数输出模块的位长

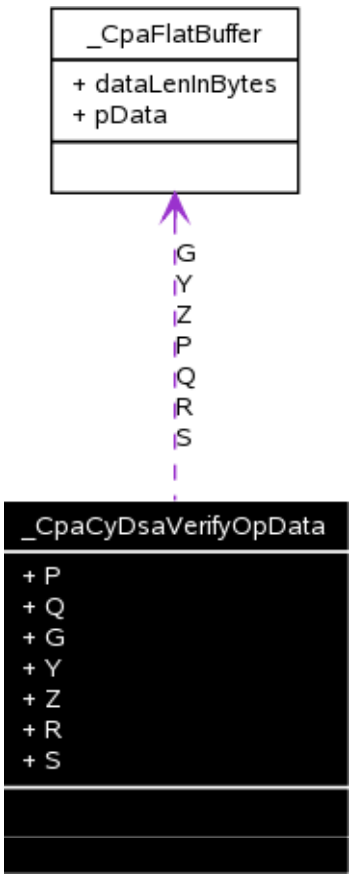
- M is the message to be signed
- m 是要签名的消息

13.5.8 _CpaCyDsaVerifyOpData Struct Reference

13.5.9 _CpaCyDsaVerifyOpData 结构引用

Collaboration diagram for _CpaCyDsaVerifyOpData:

_CpaCyDsaVerifyOpData 的协作图:



13.5.9.1 Detailed Description
13.5.9.2 详细描述

DSA Verify Operation Data.
DSA 验证操作数据。

This structure contains the operation data for the cpaCyDsaVerify 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.

13.5.7 CpaCyDsaVerifyOpData Struct

此结构包含 cpaCyDsaVerify 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 SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. P.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 P.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 cpaCyDsaVerify 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 cpaCyDsaVerify 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：

cpaCyDsaVerify()
cpaCyDsaVerify()

13.5.9.3 Data Fields

13.5.9.4 数据字段

- **CpaFlatBuffer P**
- CpaFlatBuffer P
- **CpaFlatBuffer Q**
- CpaFlatBuffer Q
- **CpaFlatBuffer G**
- CpaFlatBuffer G
- **CpaFlatBuffer Y**
- CpaFlatBuffer Y
- **CpaFlatBuffer Z**
- CpaFlatBuffer Z
- **CpaFlatBuffer R**
- CpaFlatBuffer R
- **CpaFlatBuffer S**
- CpaFlatBuffer S

13.5.9.5 Field Documentation
13.5.9.6 现场文件

DSA group parameter p
DSA 组参数 p

DSA group parameter q
DSA 组参数 q

DSA group parameter g
DSA 组参数 g

DSA public key y
DSA 公钥 y

The leftmost min(N, outlen) bits of Hash(M'), where:
Hash(M')的最左边的 min(N, outlen)位，其中:

- N is the bit length of q
- n 是 q 的比特长度
- outlen is the bit length of the hash function output block
- outlen 是哈希函数输出模块的位长
- M is the message to be signed
- m 是要签名的消息

DSA message signature r
DSA 消息签名 r

13.5.10 _CpaCyDsaStats Struct Reference

13.5.11 _CpaCyDsaStats 结构引用

13.5.11.1 Detailed

Description Cryptographic

DSA Statistics. **Deprecated:**

13.5.11.2 加密 DSA 统

计。Deprecated:

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by **CpaCyDsaStats64**.
从 Crypto API 的 1.3 版开始，这种结构已被取代，由 CpaCyDsaStats64

This structure contains statistics on the Cryptographic DSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.
此结构包含有关加密 DSA 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。

13.5.11.3 Data Fields

13.5.11.4 数据字段

- **Cpa32U numDsaPParamGenRequests**
- Cpa32U numDsaPParamGenRequests
- **Cpa32U numDsaPParamGenRequestErrors**
- Cpa32U numDsaPParamGenRequestErrors
- **Cpa32U numDsaPParamGenCompleted**
- Cpa32U numDsaPParamGenCompleted
- **Cpa32U numDsaPParamGenCompletedErrors**
- Cpa32U numDsaPParamGenCompletedErrors
- **Cpa32U numDsaGParamGenRequests**
- Cpa32U numDsaGParamGenRequests
- **Cpa32U numDsaGParamGenRequestErrors**
- Cpa32U numDsaGParamGenRequestErrors
- **Cpa32U numDsaGParamGenCompleted**
- Cpa32U numDsaGParamGenCompleted
- **Cpa32U numDsaGParamGenCompletedErrors**
- Cpa32U numDsaGParamGenCompletedErrors
- **Cpa32U numDsaYParamGenRequests**
- Cpa32U numDsaYParamGenRequests
- **Cpa32U numDsaYParamGenRequestErrors**
- Cpa32U numDsaYParamGenRequestErrors
- **Cpa32U numDsaYParamGenCompleted**
- Cpa32U numDsaYParamGenCompleted
- **Cpa32U numDsaYParamGenCompletedErrors**
- Cpa32U numDsaYParamGenCompletedErrors
- **Cpa32U numDsaRSignRequests**
- Cpa32U numDsaRSignRequests
- **Cpa32U numDsaRSignRequestErrors**
- Cpa32U numDsaRSignRequestErrors
- **Cpa32U numDsaRSignCompleted**
- Cpa32U numDsaRSignCompleted
- **Cpa32U numDsaRSignCompletedErrors**
- Cpa32U numDsaRSignCompletedErrors
- **Cpa32U numDsaSSignRequests**
- Cpa32U numDsaSSignRequests
- **Cpa32U numDsaSSignRequestErrors**
- Cpa32U numDsaSSignRequestErrors
- **Cpa32U numDsaSSignCompleted**
- Cpa32U numDsaSSignCompleted
- **Cpa32U numDsaSSignCompletedErrors**
- Cpa32U numDsaSSignCompletedErrors
- **Cpa32U numDsaRSSignRequests**
- Cpa32U numDsaRSSignRequests
- **Cpa32U numDsaRSSignRequestErrors**
- Cpa32U numDsaRSSignRequestErrors
- **Cpa32U numDsaRSSignCompleted**
- Cpa32U numDsaRSSignCompleted
- **Cpa32U numDsaRSSignCompletedErrors**
- Cpa32U numDsaRSSignCompletedErrors
- **Cpa32U numDsaVerifyRequests**
- Cpa32U numDsaVerifyRequests
- **Cpa32U numDsaVerifyRequestErrors**
- Cpa32U numDsaVerifyRequestErrors
- **Cpa32U numDsaVerifyCompleted**

- Cpa32U numDsaVerifyCompleted
- **Cpa32U numDsaVerifyCompletedErrors**
- Cpa32U numDsaVerifyCompletedErrors
- **Cpa32U numDsaVerifyFailures**
- Cpa32U numDsaVerifyFailures

13.5.11.5 Field Documentation

13.5.11.6 现场文件

Cpa32U CpaCyDsaStats::numDsaPParamGenRequests

Total number of successful DSA P parameter generation requests.
成功的 DSA P 参数生成请求的总数。

Cpa32U CpaCyDsaStats::numDsaPParamGenRequestErrors

Total number of DSA P parameter generation requests that had an error and could not be processed.
出现错误且无法处理的 DSA P 参数生成请求的总数。

Cpa32U CpaCyDsaStats::numDsaPParamGenCompleted

Total number of DSA P parameter generation operations that completed successfully.
成功完成的 DSA P 参数生成操作的总数。

Cpa32U CpaCyDsaStats::numDsaPParamGenCompletedErrors

Total number of DSA P parameter generation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 DSA P 参数生成操作的总数。

Cpa32U CpaCyDsaStats::numDsaGParamGenRequests

Total number of successful DSA G parameter generation requests.
成功的 DSA G 参数生成请求的总数。

Cpa32U CpaCyDsaStats::numDsaGParamGenRequestErrors

Total number of DSA G parameter generation requests that had an error and could not be processed.
出现错误且无法处理的 DSA G 参数生成请求的总数。

12.5.8 CpqCyDsaStats Struct

CpqCyDsaStats.numDsaGParamGenCompleted

Total number of DSA G parameter generation operations that completed successfully.
成功完成的 DSA G 参数生成操作的总数。

CpqCyDsaStats.numDsaGParamGenCompletedErrors

Total number of DSA G parameter generation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 DSA G 参数生成操作的总数。

CpqCyDsaStats.numDsaYParamGenRequests

Total number of successful DSA Y parameter generation requests.
成功的 DSA Y 参数生成请求的总数。

CpqCyDsaStats.numDsaYParamGenRequestErrors

Total number of DSA Y parameter generation requests that had an error and could not be processed.
出现错误且无法处理的 DSA Y 参数生成请求的总数。

CpqCyDsaStats.numDsaYParamGenCompleted

Total number of DSA Y parameter generation operations that completed successfully.
成功完成的 DSA Y 参数生成操作的总数。

CpqCyDsaStats.numDsaYParamGenCompletedErrors

Total number of DSA Y parameter generation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 DSA Y 参数生成操作的总数。

CpqCyDsaStats.numDsaRSignRequests

Total number of successful DSA R sign generation requests.
成功的 DSA R 签名生成请求的总数。

CpqCyDsaStats.numDsaRSignRequestErrors

Total number of DSA R sign requests that had an error and could not be processed.
出错且无法处理的 DSA R sign 请求的总数。

CpqCyDsaStats.numDsaRSignCompleted

Total number of DSA R sign operations that completed successfully.
成功完成的 DSA 签名操作的总数。

CpqCyDsaStats.numDsaRSignCompletedErrors

Total number of DSA R sign operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 DSA 签名操作的总数。

CpqCyDsaStats.numDsaSSignRequests

Total number of successful DSA S sign generation requests.
成功的 DSA 签名生成请求的总数。

CpqCyDsaStats.numDsaSSignRequestErrors

Total number of DSA S sign requests that had an error and could not be processed.
出错且无法处理的 DSA 签名请求的总数。

CpqCyDsaStats.numDsaSSignCompleted

Total number of DSA S sign operations that completed successfully.
成功完成的 DSA 签名操作的总数。

CpqCyDsaStats.numDsaSSignCompletedErrors

Total number of DSA S sign operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 DSA 签名操作的总数。

CpqCyDsaStats.numDsaRSSignRequests

Total number of successful DSA RS sign generation requests.
成功的 DSA RS 签名生成请求的总数。

Total number of DSA RS sign requests that had an error and could not be processed.
出错且无法处理的 DSA RS 签名请求的总数。

13.5.9 CpaCyDsaStats64 Struct

Cpa32U CpaCyDsaStats::numDsaRSSignCompleted

Total number of DSA RS sign operations that completed successfully.
成功完成的 DSA RS 签名操作的总数。

Total number of DSA RS sign operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 DSA RS 签名操作的总数。

Total number of successful DSA verify generation requests.
成功的 DSA 验证生成请求的总数。

Total number of DSA verify requests that had an error and could not be processed.
出错且无法处理的 DSA 验证请求的总数。

Total number of DSA verify operations that completed successfully.
成功完成的 DSA 验证操作的总数。

Total number of DSA verify operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 DSA 验证操作的总数。

Total number of DSA verify operations that executed successfully but the outcome of the test was that the verification failed. Note that this does not indicate an error.
成功执行但测试结果为验证失败的 DSA 验证操作的总数。请注意，这并不表示有错误。

13.5.12 _CpaCyDsaStats64 Struct Reference

13.5.13 _CpaCyDsaStats64 结构引用

13.5.13.1 Detailed Description
13.5.13.2 详细描述

Cryptographic DSA Statistics (64-bit version).
加密 DSA 统计信息(64 位版本)。

This structure contains 64-bit version of the statistics on the Cryptographic DSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.
此结构包含 64 位版本的加密 DSA 操作统计信息。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。

13.5.13.3 Data Fields
13.5.13.4 数据字段

- **Cpa64U numDsaPParamGenRequests**
- Cpa64U numDsaPParamGenRequests

- **Cpa64U numDsaPParamGenRequestErrors**
- Cpa64U numDsaPParamGenRequestErrors
- **Cpa64U numDsaPParamGenCompleted**
- Cpa64U numDsaPParamGenCompleted
- **Cpa64U numDsaPParamGenCompletedErrors**
- Cpa64U numDsaPParamGenCompletedErrors
- **Cpa64U numDsaGParamGenRequests**
- Cpa64U numDsaGParamGenRequests
- **Cpa64U numDsaGParamGenRequestErrors**
- Cpa64U numDsaGParamGenRequestErrors
- **Cpa64U numDsaGParamGenCompleted**
- Cpa64U numDsaGParamGenCompleted
- **Cpa64U numDsaGParamGenCompletedErrors**
- Cpa64U numDsaGParamGenCompletedErrors
- **Cpa64U numDsaYParamGenRequests**
- Cpa64U numDsaYParamGenRequests
- **Cpa64U numDsaYParamGenRequestErrors**
- Cpa64U numDsaYParamGenRequestErrors
- **Cpa64U numDsaYParamGenCompleted**
- Cpa64U numDsaYParamGenCompleted
- **Cpa64U numDsaYParamGenCompletedErrors**
- Cpa64U numDsaYParamGenCompletedErrors
- **Cpa64U numDsaRSignRequests**
- Cpa64U numDsaRSignRequests
- **Cpa64U numDsaRSignRequestErrors**
- Cpa64U numDsaRSignRequestErrors
- **Cpa64U numDsaRSignCompleted**
- Cpa64U numDsaRSignCompleted
- **Cpa64U numDsaRSignCompletedErrors**
- Cpa64U numDsaRSignCompletedErrors
- **Cpa64U numDsaSSignRequests**
- Cpa64U numDsaSSignRequests
- **Cpa64U numDsaSSignRequestErrors**
- Cpa64U numDsaSSignRequestErrors
- **Cpa64U numDsaSSignCompleted**
- Cpa64U numDsaSSignCompleted

- **Cpa64U numDsaSSignCompletedErrors**

- Cpa64U numDsaSSignCompletedErrors
- **Cpa64U numDsaRSSignRequests**
- Cpa64U numDsaRSSignRequests
- **Cpa64U numDsaRSSignRequestErrors**
- Cpa64U numDsaRSSignRequestErrors
- **Cpa64U numDsaRSSignCompleted**
- Cpa64U numDsaRSSignCompleted
- **Cpa64U numDsaRSSignCompletedErrors**
- Cpa64U numDsaRSSignCompletedErrors
- **Cpa64U numDsaVerifyRequests**
- Cpa64U numDsaVerifyRequests
- **Cpa64U numDsaVerifyRequestErrors**
- Cpa64U numDsaVerifyRequestErrors
- **Cpa64U numDsaVerifyCompleted**
- Cpa64U numDsaVerifyCompleted
- **Cpa64U numDsaVerifyCompletedErrors**
- Cpa64U numDsaVerifyCompletedErrors
- **Cpa64U numDsaVerifyFailures**
- Cpa64U numDsaVerifyFailures

13.5.13.5 Field Documentation
13.5.13.6 现场文件

Cpa64U CpaCyDsaStats64::numDsaPParamGenRequests

Total number of successful DSA P parameter generation requests.
成功的 DSA P 参数生成请求的总数。

Cpa64U CpaCyDsaStats64::numDsaPParamGenRequestErrors

Total number of DSA P parameter generation requests that had an error and could not be processed.
出现错误且无法处理的 DSA P 参数生成请求的总数。

Cpa64U CpaCyDsaStats64::numDsaPParamGenCompleted

Total number of DSA P parameter generation operations that completed successfully.
成功完成的 DSA P 参数生成操作的总数。

Cpa64U CpaCyDsaStats64::numDsaPParamGenCompletedErrors

Total number of DSA P parameter generation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 DSA P 参数生成操作的总数。

Cpa64U CpaCyDsaStats64::numDsaGParamGenRequests

Total number of successful DSA G parameter generation requests.
成功的 DSA G 参数生成请求的总数。

Cpa64U CpaCyDsaStats64::numDsaGParamGenRequestErrors

Total number of DSA G parameter generation requests that had an error and could not be processed.
出现错误且无法处理的 DSA G 参数生成请求的总数。

Cpa64U CpaCyDsaStats64::numDsaGParamGenCompleted

Total number of DSA G parameter generation operations that completed successfully.
成功完成的 DSA G 参数生成操作的总数。

Cpa64U CpaCyDsaStats64::numDsaGParamGenCompletedErrors

Total number of DSA G parameter generation operations that could not be completed successfully due to

errors.
由于错误而无法成功完成的 DSA G 参数生成操作的总数。

Cpa64U numDsaYParamGenRequests
Total number of successful DSA Y parameter generation requests.
成功的 DSA Y 参数生成请求的总数。

Cpa64U numDsaYParamGenRequestErrors
Total number of DSA Y parameter generation requests that had an error and could not be processed.
出现错误且无法处理的 DSA Y 参数生成请求的总数。

Cpa64U numDsaYParamGenCompleted
Total number of DSA Y parameter generation operations that completed successfully.
成功完成的 DSA Y 参数生成操作的总数。

Cpa64U numDsaYParamGenCompletedErrors
Total number of DSA Y parameter generation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 DSA Y 参数生成操作的总数。

CpaUint64 CpaCyDcStats64::numDcRSignRequests

Total number of successful DSA R sign generation requests.
成功的 DSA R 签名生成请求的总数。

CpaUint64 CpaCyDcStats64::numDcRSignRequestErrors

Total number of DSA R sign requests that had an error and could not be processed.
出错且无法处理的 DSA R sign 请求的总数。

CpaUint64 CpaCyDcStats64::numDcRSignCompleted

Total number of DSA R sign operations that completed successfully.
成功完成的 DSA 签名操作的总数。

CpaUint64 CpaCyDcStats64::numDcRSignCompletedErrors

Total number of DSA R sign operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 DSA 签名操作的总数。

CpaUint64 CpaCyDcStats64::numDcSSignRequests

Total number of successful DSA S sign generation requests.
成功的 DSA 签名生成请求的总数。

CpaUint64 CpaCyDcStats64::numDcSSignRequestErrors

Total number of DSA S sign requests that had an error and could not be processed.
出错且无法处理的 DSA 签名请求的总数。

CpaUint64 CpaCyDcStats64::numDcSSignCompleted

Total number of DSA S sign operations that completed successfully.
成功完成的 DSA 签名操作的总数。

CpaUint64 CpaCyDcStats64::numDcSSignCompletedErrors

Total number of DSA S sign operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 DSA 签名操作的总数。

CpaUint64 CpaCyDcStats64::numDcRSSignRequests

Total number of successful DSA RS sign generation requests.
成功的 DSA RS 签名生成请求的总数。

CpaUint64 CpaCyDcStats64::numDcRSSignRequestErrors

Total number of DSA RS sign requests that had an error and could not be processed.
出错且无法处理的 DSA RS 签名请求的总数。

CpaUint64 CpaCyDcStats64::numDcRSSignCompleted

Total number of DSA RS sign operations that completed successfully.
成功完成的 DSA RS 签名操作的总数。

CpaUint64 CpaCyDcStats64::numDcRSSignCompletedErrors

Total number of DSA RS sign operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 DSA RS 签名操作的总数。

CpaUint64 CpaCyDcStats64::numDcVerifyRequests

Total number of successful DSA verify generation requests.
成功的 DSA 验证生成请求的总数。

CpaUint64 CpaCyDcStats64::numDcVerifyRequestErrors

Total number of DSA verify requests that had an error and could not be processed.
出错且无法处理的 DSA 验证请求的总数。

CpaUint64 CpaCyDcStats64::numDcVerifyCompleted

Total number of DSA verify operations that completed successfully.

成功完成的 DSA 验证操作的总数。

Total number of DSA verify operations that could not be completed successfully due to errors.
由于错误而无法成功完成的 DSA 验证操作的总数。

Total number of DSA verify operations that executed successfully but the outcome of the test was that the verification failed. Note that this does not indicate an error.

成功执行但测试结果为验证失败的 DSA 验证操作的总数。请注意，这并不表示有错误。

13.6 Typedef Documentation

13.7 Typedef 文档

typedef struct CpaCyDsaPParamGenOpData CpaCyDsaPParamGenOpData

DSA P Parameter Generation Operation Data.

DSA P 参数生成操作数据。

This structure contains the operation data for the cpaCyDsaGenPParam 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 cpaCyDsaGenPParam 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. X.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 X.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 cpaCyDsaGenPParam 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 cpaCyDsaGenPParam 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：

cpaCyDsaGenPParam()

cpaCyDsaGenPParam()

typedef struct CpaCyDsaGParamGenOpData CpaCyDsaGParamGenOpData

DSA G Parameter Generation Operation Data.

DSA G 参数生成操作数据。

This structure contains the operation data for the cpaCyDsaGenGParam 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 cpaCyDsaGenGParam 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。

All values in this structure are required to be in Most Significant Byte first order, e.g. P.pData[0] = MSB.

All numbers MUST be stored in big-endian order.

该结构中的所有值都要求以最高有效字节优先，例如 P.pData[0] = MSB。所有数字都必须以大端顺序存储。

Note:
注意：

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDsaGenGParam 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 cpaCyDsaGenGParam 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：

cpaCyDsaGenGParam()
cpaCyDsaGenGParam()

typedef struct _CpaCyDsaYParamGenOpData CpaCyDsaYParamGenOpData;

DSA Y Parameter Generation Operation Data.
DSA Y 参数生成操作数据。

This structure contains the operation data for the cpaCyDsaGenYParam 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 cpaCyDsaGenYParam 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 SHOULD be 8-byte aligned.

为了获得最佳性能，所有数据都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. P.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 P.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 cpaCyDsaGenYParam 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 cpaCyDsaGenYParam 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：

cpaCyDsaGenYParam()
cpaCyDsaGenYParam()

typedef struct _**CpaCyDsaRSignOpData CpaCyDsaRSignOpData**
typedef 结构_CpaCyDsaRSignOpDatCpaCyDsaRSignOpDat

DSA R Sign Operation Data.
DSA R 标志操作数据。

This structure contains the operation data for the cpaCyDsaSignR 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.
该结构包含了 cpaCyDsaSignR 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 SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. P.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 P.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 cpaCyDsaSignR 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 cpaCyDsaSignR 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:

cpaCyDsaSignR()
cpaCyDsaSignR()

typedef struct _**CpaCyDsaSSignOpData CpaCyDsaSSignOpData**
typedef 结构_CpaCyDsaSSignOpDatCpaCyDsaSSignOpDat

DSA S Sign Operation Data.
DSA 的签名操作数据。

This structure contains the operation data for the cpaCyDsaSignS 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 cpaCyDsaSignS 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 SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. Q.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 Q.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 cpaCyDsaSignS 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 cpaCyDsaSignS 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:

cpaCyDsaSignS()
cpaCyDsaSignS()

typedef struct _**CpaCyDsaRSSignOpData CpaCyDsaRSSignOpData**
typedef 结构_CpaCyDsaRSSignOpDatCpaCyDsaRSSignOpDat

DSA R & S Sign Operation Data.
DSA R & S 标志操作数据。

This structure contains the operation data for the cpaCyDsaSignRS 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.

该结构包含了 cpaCyDsaSignRS 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 SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. P.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 P.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 cpaCyDsaSignRS 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 cpaCyDsaSignRS 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：

cpaCyDsaSignRS()
cpaCyDsaSignRS()

typedef struct **CpaCyDsaVerifyOpData CpaCyDsaVerifyOpData**

DSA Verify Operation Data.
DSA 验证操作数据。

This structure contains the operation data for the cpaCyDsaVerify 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 cpaCyDsaVerify 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 SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. P.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 P.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 cpaCyDsaVerify 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 cpaCyDsaVerify 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：
cpaCyDsaVerify()
cpaCyDsaVerify()

typedef struct CpaCyDsaStats CPA_DEPRECATED

Cryptographic DSA Statistics.
加密 DSA 统计。

Deprecated:
Deprecated:
As of v1.3 of the Crypto API, this structure has been deprecated, replaced by **CpaCyDsaStats64**.
从 Crypto API 的 1.3 版开始，这种结构已被取代，由 CpaCyDsaStats64

This structure contains statistics on the Cryptographic DSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.
此结构包含有关加密 DSA 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 CpaCyDsaStats64 CpaCyDsaStats64

Cryptographic DSA Statistics (64-bit version).
加密 DSA 统计信息(64 位版本)。

This structure contains 64-bit version of the statistics on the Cryptographic DSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.
此结构包含 64 位版本的加密 DSA 操作统计信息。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 void(* **CpaCyDsaGenCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpData,
typedef void(* CpaCyDsaGenCbFunc)(void *pCallbackTag, CpaStatus status, void *pOpData,
CpaBoolean

Definition of a generic callback function invoked for a number of the DSA API functions..

This is the prototype for the cpaCyDsaGenCbFunc callback function.

为许多 DSA API 函数调用的通用回调函数的定义..这是 cpaCyDsaGenCbFunc 回调函数的原型。

Context:
背景:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.
这个回调函数可以在不允许休眠发生的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

[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] *status* Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
[in] status 操作的状态。有效值为 CPA_STATUS_SUCCESS、CPA_STATUS_FAIL 和 CPA_STATUS_UNSUPPORTED。
[in] *pOpData* Opaque pointer to Operation data supplied in request.
[in] *protocolStatus* The result passes/fails the DSA protocol related checks.
[in] *pOut* Output data from the request.
[in]指向请求中提供的操作数据的 pOpData Opaque 指针。[in] protocolStatus 结果通过/未通过 DSA 协议相关检查。[in]从请求中输出数据。

Return values:
返回值:

None

没有人

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:

cpaCyDsaGenPParam() cpaCyDsaGenGParam() cpaCyDsaSignR() cpaCyDsaSignS()

cpaCyDsaGenPParam() cpaCyDsaGenGParam() cpaCyDsaSignR() cpaCyDsaSignS()

Definition of callback function invoked for cpaCyDsaSignRS requests.

为 cpaCyDsaSignRS 请求调用的回调函数的定义。

typedef void(* **CpaCyDsaRSSignCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpData,

typedef void(*CpaCyDsaRSSignCbFunc)(void *pCallbackTag, CpaStatus status, void *pOpData,

This is the prototype for the cpaCyDsaSignRS callback function, which will provide the DSA message signature r and s parameters.

这是 cpaCyDsaSignRS 回调函数的原型，它将提供 DSA 消息签名 r 和 s 参数。

Context:

背景:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.

这个回调函数可以在不允许休眠发生的上下文中执行。

Assumptions:

假设:

None

没有人

Side-Effects:

副作用:
None
没有人

Reentrant:
可重入:
No
不

Thread-safe:
线程安全:
Yes
是

Parameters:
参数:
[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] *status* Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
[in] status 操作的状态。有效值为 CPA_STATUS_SUCCESS、CPA_STATUS_FAIL 和 CPA_STATUS_UNSUPPORTED。
[in] *pOpData* Operation data pointer supplied in request.
[in]请求中提供了 pOpData 操作数据指针。
[in] *protocolStatus* The result passes/fails the DSA protocol related checks.
[in] protocolStatus 结果通过/未通过 DSA 协议相关检查。
[in] *pR* DSA message signature r.
[in] pR DSA 消息签名
[in] *pS* DSA message signature s.
pS DSA 消息签名。

Return values:
返回值:
None
没有人

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:

cpaCyDsaSignRS()
cpaCyDsaSignRS()

Definition of callback function invoked for cpaCyDsaVerify requests.

typedef void(* **CpaCyDsaVerifyCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpData,

typedef void(* CpaCyDsaVerifyCbFunc)(void *pCallbackTag, CpaStatus status, void *pOpData,

This is the prototype for the cpaCyDsaVerify callback function.

为 cpaCyDsaVerify 请求调用的回调函数的定义。这是

cpaCyDsaVerify 回调函数的原型。

Context:
背景:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.
这个回调函数可以在不允许休眠发生的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

[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] *status* Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.

[in] status 操作的状态。有效值为 CPA_STATUS_SUCCESS、CPA_STATUS_FAIL 和 CPA_STATUS_UNSUPPORTED。

[in] *pOpData* Operation data pointer supplied in request.

[in]请求中提供了 pOpData 操作数据指针。

[in] *verifyStatus* The verification passed or failed.

[in] verifyStatus 验证通过或失败。

Return values:
返回值:

None
没有人

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:

cpaCyDsaVerify()
cpaCyDsaVerify()

13.7 Function Documentation

13.7 功能文件

```
cpaCyDsaGenPParam (  const                                    instanceHandle, pCb,
const
                                    void *pCallbackTag,
                                    const *  pOpData,
                                    *pProtocolStatus,
                                    *pP
                                  )
```

```
cpaCyDsaGenPParam (  const                                          instanceHandle, pCb,
const
                                void *pCallbackTag,
                                const * pOpData,
                                *pProtocolStatus,
                                *pP
                               )
```

Generate DSA P Parameter.
生成 DSA P 参数。

This function performs FIPS 186-3 Appendix A.1.1.2 steps 11.4 and 11.5, and part of step 11.7:
该功能执行 FIPS 186-3 附录 A.1.1.2 步骤 11.4 和 11.5，以及步骤 11.7 的一部分:

11.4. c = X mod 2q. 11.5. p = X - (c - 1). 11.7. Test whether or not p is prime as specified in Appendix C.3. [Note that a GCD test against ~1400 small primes is performed on p to eliminate ~94% of composites - this is NOT a "robust" primality test, as specified in Appendix C.3.]
11.4.c = X mod 2q。11.5.p = X - (c - 1)。11.7.按照附录 C.3 中的规定，测试 p 是否为素数。[注意，对 p 进行了针对约 1400 个小素数的 GCD 测试，以消除约 94%的复合数-这不是一个“稳健”的素性测试，如附录 C.3 中的规定。]

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 p is in the right range and has passed the limited primality test.
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 p 是否在正确的范围内，以及是否通过了有限的素性测试。

Specifically, (protocolStatus == CPA_TRUE) means p is in the right range and SHOULD be subjected to a robust primality test as specified in FIPS 186-3 Appendix C.3 (for example, 40 rounds of Miller-Rabin).
具体来说，(protocolStatus == CPA_TRUE)意味着 p 在正确的范围内，并且应该接受 FIPS 186-3 附录 C.3 中规定的稳健素性测试(例如，40 轮米勒-拉宾测试)。
Meanwhile, (protocolStatus == CPA_FALSE) means p is either composite, or p < 2^(L-1), in which case the value of p gets set to zero.
同时，(protocolStatus == CPA_FALSE)意味着 p 是复合的，或者 p < 2^(L-1)，在这种情况下，p 的值被设置为零。

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