image

**东软睿驰秘密(Neusoft Reach Secret)**

文件编号(File No.)：项目编号- D00-COP02-SW-T31-流水号

NeuSAR aCore

软件需求规格说明书

(Software requirement specification)

**(Crypto)**

版本(Version)：0.7

日期(Date)：2022-04-18

|  |  |  |
| --- | --- | --- |
| 编制  Prepared by | 审核  Reviewed by | 批准  Approved by |
|  |  |  |

东软睿驰汽车技术(沈阳)有限公司

**(版权所有，翻版必究)**

Neusoft Reach Automotive Technology(Shenyang) Co., Ltd

**(Copyright by Reach Corporation, All Rights Reserved)**

变更履历(Change Log)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **修改编号**  **No.** | **版本**  **Version** | **修改内容**  **Contents Revised** | **状态**  **Status** | **修改人/日期**  **Reviser/Date** | **审批人/日期**  **Approve/Date** |
| 1 | 0.7 | 初稿 | Draft | 黄海军、秦天宇/2022/4/18 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Contents**

[1 引言(Introduction) 1](#_Toc97295388)

[1.1 目的(Goal) 1](#_Toc97295389)

[1.2 范围(Scope) 1](#_Toc97295390)

[1.3 参考文档(Reference) 1](#_Toc97295391)

[1.4 术语和缩略语(Terms And Abbreviations) 1](#_Toc97295392)

[2 软件系统概述(Software System Overview) 2](#_Toc97295393)

[2.1 软件系统背景(Software System Background) 2](#_Toc97295394)

[2.2 软件系统目标(Software System Goal) 2](#_Toc97295395)

[2.3 外部关联 (External Association) 2](#_Toc97295396)

[3 功能需求(Functional Requirement) 2](#_Toc97295397)

[3.1 功能需求1(Function Requirement No.1) 2](#_Toc97295398)

[3.1.1 [SWRD-ID]功能子需求1-1(Function Requirement No.1-1) 2](#_Toc97295399)

[3.1.2 [SWRD-ID]功能子需求1-2(Function Requirement No.1-2) 3](#_Toc97295400)

[3.2 功能需求2(Function Requirement No.2) 3](#_Toc97295401)

[3.2.1 [SWRD-ID]功能子需求2-1(Function Requirement No.2-1) 3](#_Toc97295402)

[3.2.2 [SWRD-ID]功能子需求2-2(Function Requirement No.2-2) 4](#_Toc97295403)

[4 非功能需求(Non-Functional Requirements) 4](#_Toc97295404)

[4.1 制约(Constraint) 4](#_Toc97295405)

[4.1.1 [SWRD-ID]非功能需求1(Non-Function Requirement No.1) 4](#_Toc97295406)

[4.1.2 [SWRD-ID]非功能需求2(Non-Function Requirement No.2) 5](#_Toc97295407)

[4.2 性能质量要求(Performance Quality Requirements) 5](#_Toc97295408)

[4.2.1 [SWRD-ID]非功能需求3(Non-Function Requirement No.3) 5](#_Toc97295409)

[4.2.2 [SWRD-ID]非功能需求4(Non-Function Requirement No.4) 5](#_Toc97295410)

[5 接口说明(API) 6](#_Toc97295411)

[5.1 接口头文件(API Header files) 6](#_Toc97295412)

[5.2 接口共同数据类型(API Common Data Types) 6](#_Toc97295413)

[5.2.1 [SWRD-ID]数据类型1(Data Type1) 6](#_Toc97295414)

[5.2.2 [SWRD-ID]数据类型2(Data Type2) 7](#_Toc97295415)

[5.3 接口定义(API Reference) 7](#_Toc97295416)

[5.3.1 [SWRD-ID]接口1(API 1) 7](#_Toc97295417)

[5.3.2 [SWRD-ID]接口2(API 2) 8](#_Toc97295418)

[附录A- 信息定义 9](#_Toc97295419)

[附录B- 配置信息 11](#_Toc97295420)

# 引言(Introduction)

## 目的(Goal)

为idsm模块提供详细的软件需求的定义，给开发人员和测试人员提供设计和测试执行的标准。

## 范围(Scope)

本文使用者： 开发人员、测试人员、PSM、TeamLeader、TestLeader和产品负责人。

本文使用方法：

* 对于开发人员、根据本文档中定义的功能/非功能要求进行后续的设计。
* 对于测试人员、通过理解本文档中的要求，进行测试用例的制作和后续测试执行。

对于PSM、TeamLeader、TestLeader和产品负责人、来判断需求理解的正确性

## 参考文档(Reference)

|  |  |  |
| --- | --- | --- |
| **编号** | **SVN路径\文档名** | **文档版本** |
| 1 | AUTOSAR\_RS\_IntrusionDetectionSystem | AP-R21-11 |
| 2 | AUTOSAR\_RS\_SecurityExtractTemplate | AP-R21-11 |
| 3 | AUTOSAR\_SWS\_AdaptiveIntrusionDetectionSystemManager | AP-R21-11 |
| 4 | AUTOSAR\_PRS\_IntrusionDetectionSystem | AP-R21-11 |
|  |  |  |
|  |  |  |

## 术语和缩略语(Terms And Abbreviations)

|  |  |  |
| --- | --- | --- |
| **编号** | **缩写** | **说明** |
| 1 | idsM | ⼊侵检测系统管理器。 |
| 2 | IDS | ⼊侵检测系统 |
| 3 | IdsR | ⼊侵检测系统报告器 |
| 4 | SEv | 安全事件 |
| 5 | QSEv | 合格的安全事件 |
| 6 | Sem | 安全事件存储器 |
| 7 | SIEM | 安全事件和事件管理 |
| 8 | SOC | 安全运营中⼼ |
| 9 | DID | 根据统⼀诊断服务的数据标识符 |
| 10 | DTC | 诊断故障代码 |
| 11 | FC | 功能集群 |
| 12 | SecXT | 安全提取 |
| 13 | SWCL | 软件集群 |

# 软件系统概述(Software System Overview)

## 软件系统背景(Software System Background)

NeuSAR aCore 架构将 NeuSAR aCore 基础的软件组织为功能集群。这些集群提供通用功能作为应用程序的服务。功能集群⼊侵检测系统管理器 是 NeuSAR aCore架构的一部分，为应用程序提供安全事件检测相关的服务。

## 软件系统目标(Software System Goal)

idsM为应用程序和其他自适应 AUTOSAR 功能集群提供了一些标准化接口，这些接口提供了系统安全功能。 这些功能主要包括识别，过滤，保存，上报安全事件等，以保证系统运行时识别相应的风险。

### CryptoProvider

FC Crypto 及其 CryptoAPI 支持公钥和对称密钥加密。它也允许应用程序使用身份验证、加密和解密等机制，以满足汽车服务中的机密性需求。

FC Crypto定义的接口在设计时要能够集成第三方加密库和硬件基本单元。当 FC Crypto 的默认加密库没有提供某些算法，或没有提供硬件加速的时，可以有效的对接其他安全“trust anchor”或硬件加速加密。

CryptoAPI 提供了一组方法，使应用程序和系统开发人员能够在存储和传输信息时保护信息免受入侵者的侵害。为了保证通信过程中关键信息的保密性或真实性，接受者身份的可靠性，FC Crypto 提供了一套安全机制，以确保达到一下安全目标：

身份验证：FC Crypto 提供允许自适应应用程序或功能集群向其他应用程序或功能集群证明其身份的机制。

不可否认性：FC Crypto 支持不可否认性的概念，即某人不能否认某事的有效性。

机密性：FC Crypto 支持将信息保密。 密码系统最初就是为了这种能力而开发的。 无论是在系统调试或跟踪期间发送的系统或用户特定数据，还是存储车辆/ECU 的机密数据，加密都可以确保只有有权访问相应密钥的用户才能获得对数据明文的读取权限。

完整性：FC Crypto 确保在存储或传输期间不会更改受保护的数据。此外，FC Crypto 允许应用程序去构建保证元素或服务的完整性的机制。

此外，FC Crypto 要保证过程的机密性，即不要泄露有关从流中读取的消息的任何信息，直到解密过程完成且没有错误。

### Key Storage Provider

FC Crypto 提供Key Storage Provider密钥存储。该provider目的是对支持的密钥对象进行安全持久存储，并通过统一接口对它们进行编程访问，使密钥对象能够独立存在于实际的物理存储中。单个逻辑密钥存储可以聚合多个基于软件或硬件的物理存储*Crypto Provider*。 这对 Key Storage Provider接口的用户是透明的。为保证正确地访问密钥，CryptoAPI 将通过IAM来限制了对密钥材料的访问。

### X.509 Provider

CryptoAPI 支持所有典型的证书（DER、PEM的格式证书）的导出和验证等基本功能。所有对证书的相关功能操作都由证书管理完成，证书管理通过IAM实施策略限制访问权限。

以上，定义了FC Crypto 的系统目标和实现。图 7.1 概括的描述了FC Crypto的整理架构， 包括用于证书管理的 X.509 Provider、Crypto Provider 和 Key Storage Provider，以及与应用程序和HSM之间调用关系。

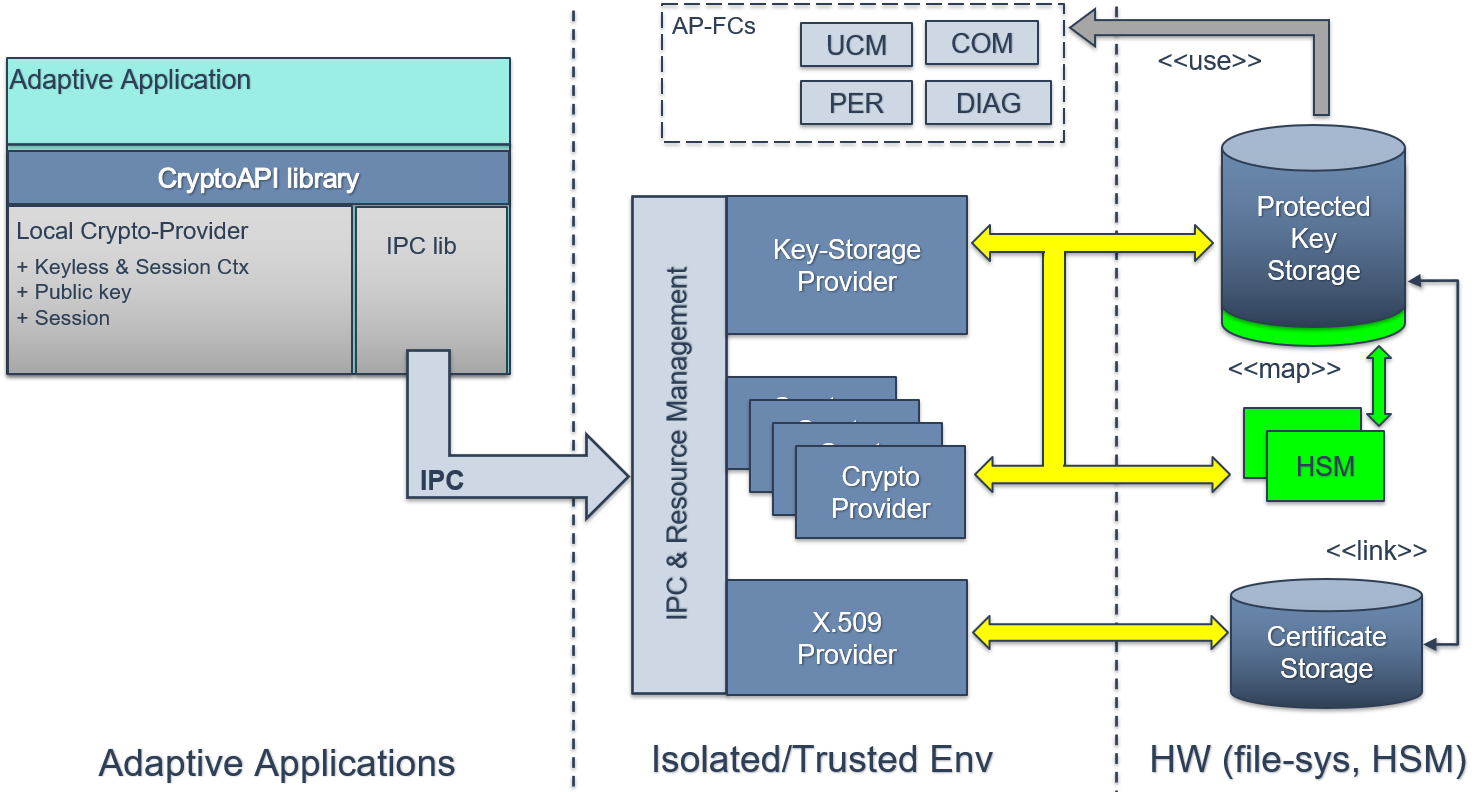


图2.1 Crypto 总体架构

## 外部关联 (External Association)

### IAM

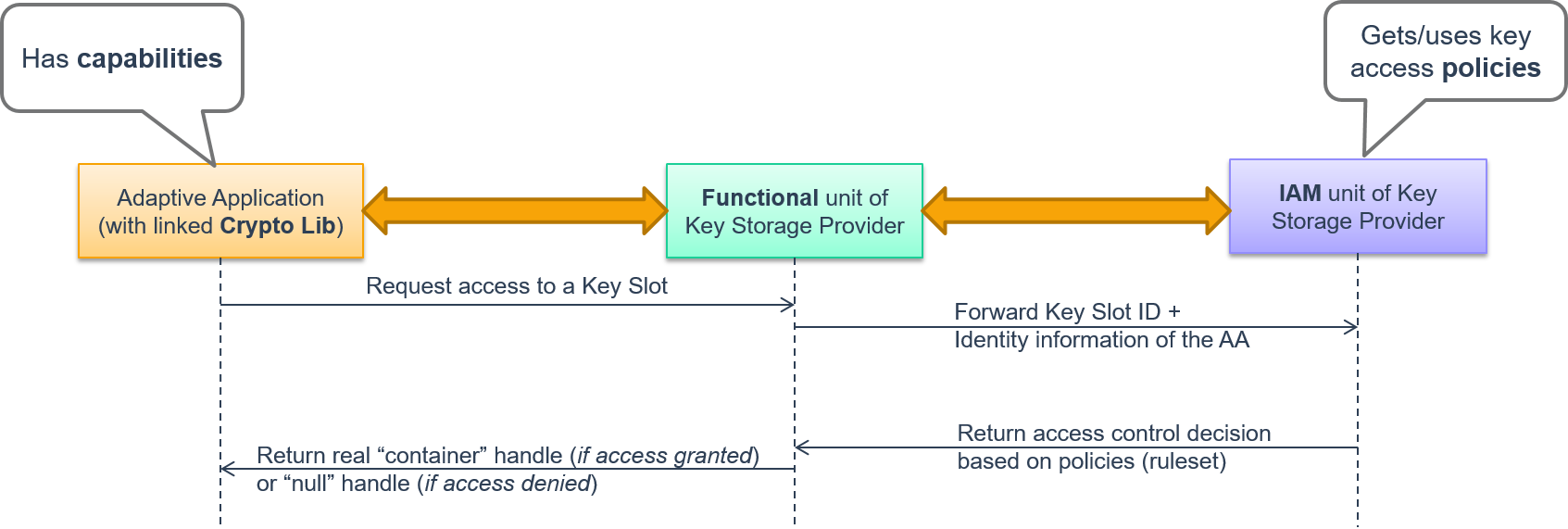


图2.2 Crypto 与 IAM的交互

IAM 启用对建模实体或资源的访问控制。 目前，FC Crypto 只考虑访问控制Key Slot（读/写）。 为了简化 IAM 配置，FC Crypto 指定了独占访问模型，该模型规定只能将对密钥槽的访问权限授予单个自适应应用程序（独占）。

# 功能需求(Functional Requirement)

## 总体结构

总体结构部分明确了加密模块的总体结构需求，包括总体结构的设计需求以及实现需求；以及对于各子模块的主要需求点。

### [SWRD-Crypto-00001] 结构设计与实现

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00001 |
| **Type** | Valid |
| **Description** | 加密模块主要包括如下：CryptoAPI CryptoStack CryptoProvider KeyStorageProvider CertificateManagementProvider；其中CryptoAPI对外以提供加密库形式提供API接口， Crypto Stack为加密模块功能的守护进程，实现了对于密码学服务、密钥管理、证书管理统一管理； Crypto API 与Crypto Stack采用IPC通信； CryptoProvider提供密码学服务，支持可以提供多CryptoProvider，可以为第三方加密库或者硬件加密服务；KeyStorageProvider提供密钥管理服务，安全存储密钥以及加密对象；必须确保处理和存储对象的机密性和真实性；CertificateManagementProvider提供X509证书的解析验证以及存储管理等服务；各Provider间相互独立； |
| **Upstream ID** | User\_defined\_00002 User\_defined\_00006 User\_defined\_00008  User\_defined\_00013 User\_defined\_00014 User\_defined\_00015 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：加密模块总体设计结构完成  主要测试点：总体结结构中是否包括了 CryptoAPI CryptoStack CryptoProvider KeyStorageProvider CertificateManagementProvider基本单元；Crypto API 与Crypto Stack 是否采用IPC通信；是否可支持集成第三方加密库或者硬件加密服务；各Provider是否提供了对应的服务；  成功标准：   1. 总体结构中包括了上述基本单元； 2. Crypto API 与Crypto Stack采用了IPC通信方式； 3. 支持集成第三方加密库或者硬件加密服务； 4. 各Provier提供了对应的服务； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00002] 密钥与证书资源访问

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00002 |
| **Type** | Valid |
| **Description** | 密钥统一管理，密钥涉及存储在软件数据库或者HSM中，对于上述密钥进行统一管理，HSM中存储的密钥与统一管理进行访问关系映射，保证能够准确的访问到HSM中密钥；  密钥权限控制， 密钥槽分为两种类型：Application 以及 Machine两种类型。密钥槽的访问基于IAM模块进行权限管理；  证书权限控制，证书槽的访问基于IAM模块进行权限管理；  密钥更新与使用并发进行，在KeyStorageProvider中UpdateObserver模块检测密钥更新事件，及时对密钥使用者进行通知； |
| **Upstream ID** | User\_defined\_00005 User\_defined\_00009 User\_defined\_00010  User\_defined\_00011 User\_defined\_00012 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：加密模块总体设计结构完成  主要测试点：密钥管理模块对于密钥是否进行了统一管理，设计中是否对于HSM中密钥关系映射到总体管理中；密钥槽的类型是否区分了Application和Machine类型；密钥的访问是否基于IAM模块进行了权限控制；证书的访问是否基于IAM模块进行了权限访问控制；密钥更新与使用并发进行，是否可进行对密钥使用者进行事件通知，同时密钥更新是否有效；  成功标准：   1. 密钥进行了统一管理，包括对于HSM中的密钥关系映射； 2. 密钥槽类型设计了上述两种类型； 3. 对于证书的访问以及密钥的访问都基于IAM模块进行了权限控制； 4. 密钥更新与使用，保证及时有效通知密钥使用者，不影响密钥的使用，同时保证密钥有效更新； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

## Crypto

Crypto模块基于密码学技术实现对称加密，非对称加密，数字签名与验签，散列计算，随机数生成，消息认证码，基于对称算法密钥包装，非对称算法密钥封装，密钥协商，密钥生成，密钥派生等功能；同时提供密钥明文或者密文导入导出接口，密钥类对象加载，加密算法ID与名称转换，获取实例化CryptoProvider，运行对象与持久化资源访问中间层接口（IOinterface）;

### [SWRD-Crypto-00100]随机数生成

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00100 |
| **Type** | Valid |
| **Description** | 在很多的加密算法中或者应用中，都会使用Salts或者Nonces，因此，根据需要将获取一定长度的随机数。随机数生成功能模块，将提供相关接口进行随机数的生成；  应用随机数生成功能流程如下：  1 创建随机数生成上下文；  2 调用随机数生成接口，生成指定长度的随机数数据； |
| **Upstream ID** | SWS\_CRYPT\_00500 SWS\_CRYPT\_00501 SWS\_CRYPT\_00502  SWS\_CRYPT\_00503 SWS\_CRYPT\_00506 SWS\_CRYPT\_00507 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：随机数生成功能实现完成  主要测试点：创建随机数上下文是否成功；调用随机数生成接口是否能够生成指定的长度的随机数；多次调用生成随机数生成，验证随机数是否随机化；接口异常测试，返回错误码是否正确；  成功标准：   1. 创建随机数上下文成功； 2. 随机数生成随机化，并且能够生成指定长度的随机数； 3. 接扣异常测试，返回错误码正确 |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 本次实现的随机数功能，实现的是全局状态随机数生成功能；标准中要实现全局状态以及局部状态； |

### [SWRD-Crypto-00101]散列计算

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00101 |
| **Type** | Valid |
| **Description** | 散列计算是密码学技术中的基本元素，散列计算是一种单向计算，将任意长度的数据转换成固定长度的数据输出；同时由于算法特性，不能逆向运算；常用的散列算法需要支持如下：SHA1 SHA256 SM3 MD5;  应用散列计算功能流程如下：  1 创建散列计算上下文；  2 初始化散列计算上下文；  3 应用新的数据更新散列计算上下文；  4 完成摘要计算；  散列计算功能接口调用，具有一定顺序，如果不能按照接口顺序调用接口，在调用接口时，会返回一定的错误码； |
| **Upstream ID** | SWS\_CRYPT\_00901 SWS\_CRYPT\_00902 SWS\_CRYPT\_00903  SWS\_CRYPT\_00905 SWS\_CRYPT\_00906 SWS\_CRYPT\_00907  SWS\_CRYPT\_00908 SWS\_CRYPT\_00909 SWS\_CRYPT\_00910  SWS\_CRYPT\_00919 User\_defined\_00018 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：散列计算功能实现完成  主要测试点：创建散列计算上下文是否成功；是否完全支持了上述的算法；基于上述流程，调用相关接口，实现散列计算，计算的结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1创建散列计算上下文成功；  2 上述散列算法全部支持；  3 基于散列计算流程，调用相应接口，散列计算结果正确；  4 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00102]对称块加密计算

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00102 |
| **Type** | Valid |
| **Description** | 对称块加密算法应用对称密钥来加解密输入数据；块加密通常用于保护静态数据，例如文件系统上的数据等；常用的对称块加密算法需要支持如下：AES-128 AES-256 SM4;  应用对称块加密功能流程如下：  1 创建对称块加密上下文；  2 初始化对称块加密上下文；  3 对数据进行加解密处理；  对称块加密功能接口调用，具有一定顺序，如果不能按照接口顺序调用接口，在调用接口时，会返回一定的错误码； |
| **Upstream ID** | SWS\_CRYPT\_40963 SWS\_CRYPT\_01501 SWS\_CRYPT\_01502  SWS\_CRYPT\_01503 SWS\_CRYPT\_01504 SWS\_CRYPT\_01506  SWS\_CRYPT\_01508 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：对称块加密功能实现完成  主要测试点：创建对称块加密上下文是否成功；是否完全支持了上述的算法；基于上述流程，调用相关接口，实现数据加解密，计算的结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1创建对称块加密上下文成功；  2 上述对称加密算法全部支持；  3 基于加解密流程，调用相应接口，数据加解密结果正确；  4 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00103]对称流加密计算

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00103 |
| **Type** | Valid |
| **Description** | 对称流加密算法应用对称密钥来加解密输入数据；流加密适用于资源消耗低的快速实现，流加密通常用于保护动态数据，例如加密网络上的数据等；常用的对称流加密算法需要支持如下：AES-128 AES-256 SM4;  应用对称块加密功能流程如下：  1 创建对称块加密上下文；  2 初始对称块加密上下文；  3 对数据进行加解密处理；  4 流加密处理结束；  对称块加密功能接口调用，具有一定顺序，如果不能按照接口顺序调用接口，在调用接口时，会返回一定的错误码； |
| **Upstream ID** | SWS\_CRYPT\_40964 SWS\_CRYPT\_01651 SWS\_CRYPT\_01653  SWS\_CRYPT\_01654 SWS\_CRYPT\_01655 SWS\_CRYPT\_01656  SWS\_CRYPT\_01657 SWS\_CRYPT\_01658 SWS\_CRYPT\_01659  SWS\_CRYPT\_01660 SWS\_CRYPT\_01661 SWS\_CRYPT\_01662  User\_defined\_00019 User\_defined\_00020 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：对称流加密功能实现完成  主要测试点：创建对称流加密上下文是否成功；是否完全支持了上述的算法；基于上述流程，调用相关接口，实现数据加解密，计算的结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1创建对称流加密上下文成功；  2 上述对称加密算法全部支持；  3 基于加解密流程，调用相应接口，数据加解密结果正确；  4 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00104]非对称加解密

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00104 |
| **Type** | Valid |
| **Description** | 非对称加解密基于一对密钥：公钥和私钥；公钥是可以分发给每个人，私钥需要自己进行安全保护；应用公钥对数据进行加密，应用私钥对数据进行解密；常用的非对称加解密算法需要支持如下：RSA SM2;  应用非对称加解密功能流程如下：  1 创建非对称加解密上下文；  2 初始非对称加解密上下文；  3 对数据进行加解密处理； |
| **Upstream ID** | SWS\_CRYPT\_40966 SWS\_CRYPT\_02700 SWS\_CRYPT\_02701  SWS\_CRYPT\_02702 SWS\_CRYPT\_02703 SWS\_CRYPT\_02704  SWS\_CRYPT\_02705 SWS\_CRYPT\_02706 SWS\_CRYPT\_02726 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：非对称加解密功能实现完成  主要测试点：创建非对称加解密上下文是否成功；是否完全支持了上述的算法；基于上述流程，调用相关接口，实现数据加解密，计算的结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1创建非对称加解密上下文成功；  2 上述非对称加密算法全部支持；  3 基于加解密流程，调用相应接口，数据加解密结果正确；  4 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00105]数字签名与验签

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00105 |
| **Type** | Valid |
| **Description** | 数字签名与验签基于非对称算法以及散列算法；应用私钥对散列计算后的数据进行签名，应用公钥钥对数据签名进行验证；常用的非对称算法需要支持如下：RSA ECC SM2;  应用数字签名与验签功能流程如下：  1 创建数字签名或验签上下文；  2 初始签名或验签上下文；  3 对数据进行签名或者验签梳理； |
| **Upstream ID** | SWS\_CRYPT\_02400 SWS\_CRYPT\_02414 SWS\_CRYPT\_01821  SWS\_CRYPT\_02417 SWS\_CRYPT\_02418 SWS\_CRYPT\_02419  SWS\_CRYPT\_02408 SWS\_CRYPT\_02413 SWS\_CRYPT\_01820  SWS\_CRYPT\_02415 SWS\_CRYPT\_02416 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：数字签名与验签功能实现完成  主要测试点：创建数字签名或者验签上下文是否成功；是否完全支持了上述的算法；基于上述流程，调用相关接口，实现数据签名或验签，结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1创建数字签名或验签上下文成功；  2 上述数字签名与验签算法全部支持；  3 基于数字签名与验签流程，调用相应接口，数字签名与验签结果正确；  4 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00106]消息签名编码与消息验签恢复

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00106 |
| **Type** | Valid |
| **Description** | 消息签名与编码应该对提供的输入缓冲区（消息）进行签名，并根据为此上下文配置的算法将消息编码为生成的签名；消息验签与恢复是应该从提供的签名中解码消息，并且只有在根据为此上下文配置的算法成功验证后才返回消息。  应用消息签名编码和消息验签恢复功能流程如下：  1 创消息签名编码或消息验签恢复上下文；  2 初始化上下文；  3 对数据进行签名编码或者验签解码处理； |
| **Upstream ID** | SWS\_CRYPT\_02409 SWS\_CRYPT\_02412 SWS\_CRYPT\_01822  SWS\_CRYPT\_02420 SWS\_CRYPT\_02410 SWS\_CRYPT\_02411  SWS\_CRYPT\_01823 SWS\_CRYPT\_02421 SWS\_CRYPT\_02422 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：消息签名编码与消息验签恢复功能实现完成  主要测试点：创建消息签名编码或消息验签恢复上下文是否成功；基于上述流程，调用相关接口，实现消息签名编码与消息验签恢复，结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1创建消息签名与编码或消息验签恢复上下文成功；  2 基于上述流程，调用相应接口，消息签名编码与消息验签恢复结果正确；  3 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00107]密钥派生

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00107 |
| **Type** | Valid |
| **Description** | 支持基于密钥派生算法进行对称密钥或者秘密种子派生；密钥派生 (KDF) 应防止攻击者在获得派生密钥时收集有关主密钥值或其他派生密钥的信息 . 加强派生密钥以防止攻击者猜测或暴力破解派生密钥也很重要。 因此，通过添加避免字典攻击的盐和增加猜测延迟的多次迭代来导出较好的密钥。常用的密钥派生算法需要支持如下：HKDF;  应用密钥派生功能流程如下：  1 创建密钥派生上下文；  2 增加盐值，配置迭代次数；  3 调用密钥派生接口生成密钥； |
| **Upstream ID** | SWS\_CRYPT\_00601 SWS\_CRYPT\_00603 SWS\_CRYPT\_00608  SWS\_CRYPT\_00609 SWS\_CRYPT\_00610 SWS\_CRYPT\_00611  SWS\_CRYPT\_40944 SWS\_CRYPT\_40945 SWS\_CRYPT\_40946 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：密钥派生功能实现完成  主要测试点：创建密钥派生上下文是否成功；是否完全支持了上述的算法；基于上述流程，调用相关接口，实现密钥派生，结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1创建密钥派生上下文成功；  2 上述密钥派生算法全部支持；  3 基于密钥派生流程，调用相应接口，密钥派生结果正确；  4 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00108]消息认证

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00108 |
| **Type** | Valid |
| **Description** | 消息认证码（MAC）保证了数据完整性以及不可伪造性；常用生成消息认证码的算法需要支持如下：HMAC CMAC;  应用消息认证码功能流程如下：  1 创建消息认证码上下文；  2 初始化上下文-设置密钥；  3 开始消息认证码计算；  4 更新消息认证码计算；  5 结束消息认证码计算；  消息认证功能接口调用，具有一定顺序，如果不能按照接口顺序调用接口，在调用接口时，会返回一定的错误码； |
| **Upstream ID** | SWS\_CRYPT\_01200 SWS\_CRYPT\_01201 SWS\_CRYPT\_01202  SWS\_CRYPT\_01203 SWS\_CRYPT\_01204 SWS\_CRYPT\_01207  SWS\_CRYPT\_01208 SWS\_CRYPT\_01209 SWS\_CRYPT\_01210  SWS\_CRYPT\_01211 SWS\_CRYPT\_01213 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：消息认证码功能实现完成  主要测试点：创建消息认证上下文是否成功；是否完全支持了上述的算法；基于上述流程，调用相关接口，实现消息认证码计算，结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1创建消息认证码上下文成功；  2 上述消息认证码生成算法全部支持；  3 基于消息认证码生成流程，调用相应接口，结果正确；  4 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00109]对称加密包装密钥对象

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00109 |
| **Type** | Valid |
| **Description** | 对称加密包装密钥对象，主要是采用对称加密算法，对密钥对象（对称密钥， 密钥种子， 非对称密钥）进行加密包装；一般用于在不安全的环境中存储密钥或者通过不安全的通道传输密钥。  应用对称加密包装密钥对象的功能流程如下：  1 创建对称加密包装密钥对象的上下文；  2 初始化上下文-设置KEK；  3 根据需要，选择对数据进行包装或者解包； |
| **Upstream ID** | SWS\_CRYPT\_40965 SWS\_CRYPT\_02121 SWS\_CRYPT\_02122  SWS\_CRYPT\_02123 SWS\_CRYPT\_02104 SWS\_CRYPT\_02105  SWS\_CRYPT\_02107 SWS\_CRYPT\_02108 SWS\_CRYPT\_02109 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：对称加密包装密钥对象功能实现完成  主要测试点：创建对称加密包装密钥对象上下文是否成功；基于上述流程，调用相关接口，对密钥对象数据进行包装或者解包，结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1创建对称加密包装密钥对象上下文成功；  2 基于对称加密包装或者解包密钥对象流程，调用相应接口，结果正确；  3 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00110]密钥封装与解封机制

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00110 |
| **Type** | Valid |
| **Description** | 密钥封装机制 (KEM) 的工作方式与公钥加密方案类似，不同之处在于加密算法除了另一个密钥之外不接受任何输入。在对密钥要进行封装时，采用非对称公钥加密；在对密钥进行解封时，采用非对称私钥解密；主要对对称密钥或者秘密种子应用；  密钥封装与密钥解封的功能流程如下：  1 创建密钥封装或者密钥解封的上下文；  2 初始化上下文；  3 根据需要，选择对数据进行封装或者解封； |
| **Upstream ID** | SWS\_CRYPT\_40967 SWS\_CRYPT\_03000 SWS\_CRYPT\_03002  SWS\_CRYPT\_03006 SWS\_CRYPT\_03007 SWS\_CRYPT\_03008  SWS\_CRYPT\_40968 SWS\_CRYPT\_03003 SWS\_CRYPT\_03004  SWS\_CRYPT\_03005 SWS\_CRYPT\_03009 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：密钥封装与解封功能实现完成  主要测试点：创建密钥封装与解封上下文是否成功；基于上述流程，调用相关接口，对密钥对象数据进行封装或者解封，结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1创建密钥封装或者解封上下文成功；  2 基于密钥封装或者解封流程，调用相应接口，结果正确；  3 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00111]密钥协商

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00111 |
| **Type** | Valid |
| **Description** | 密钥协商是指通过交换密钥材料而实际不传输密钥自身，通过密钥材料以及安全算法，两端获得同一密钥，保证了密钥的安全性。密钥交换算法：ECDH  应用密钥协商功能流程如下：  1 创建密钥协商上下文；  2 初始化上下文-设置密钥；  3 调用相关接口，生成密钥或者秘密种子； |
| **Upstream ID** | SWS\_CRYPT\_40969 SWS\_CRYPT\_03300 SWS\_CRYPT\_03311  SWS\_CRYPT\_03312 SWS\_CRYPT\_03313 SWS\_CRYPT\_03301  SWS\_CRYPT\_03302 SWS\_CRYPT\_03303 SWS\_CRYPT\_03304  User\_defined\_00022 User\_defined\_00023 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：密钥协商功能实现完成  主要测试点：创建密钥协商上下文是否成功；基于上述流程，调用相关接口，生成密钥或者秘密种子，结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1创建密钥协商上下文成功；  2 基于密钥协商生成密钥或者秘密种子流程，调用相应接口，结果正确；  3 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00112]认证加密

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00112 |
| **Type** | Valid |
| **Description** | 认证加密（AE）或者带有关联数据的认证加密（AEAD）同时提供机密性和数据真实性。支持算法：ChaCha20-Poly1305 AES-GCM  应用密钥协商功能流程如下：  1 创建认证加密上下文；  2 初始化上下文-设置密钥；  3 调用相关接口，对数据进行处理； |
| **Upstream ID** | SWS\_CRYPT\_01806 SWS\_CRYPT\_01800 SWS\_CRYPT\_01801  SWS\_CRYPT\_01802 SWS\_CRYPT\_01803 SWS\_CRYPT\_01804  SWS\_CRYPT\_01805 SWS\_CRYPT\_01807 SWS\_CRYPT\_01808  SWS\_CRYPT\_01811 User\_defined\_00021 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：认证加密功能实现完成  主要测试点：创建认证加密上下文是否成功；基于上述流程，调用相关接口，进行数据认证加密处理，结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1创建认证加密上下文成功；  2 基于认证加密数据处理流程，调用相应接口，结果正确；  3 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00113]密钥生成

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00113 |
| **Type** | Valid |
| **Description** | 在不同的加密算法中，使用的密钥类型不同，需要根据算法生成不同的密钥；常用的有秘密种子，对称密钥，非对称密钥对；Crypto模块提供上述密钥等生成接口； |
| **Upstream ID** | SWS\_CRYPT\_40962 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：密钥类生成实现完成  主要测试点：调用密钥类生成接口，验证结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1 调用密钥类生成接口，结果正确；  2 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00114]密钥明文导入导出

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00114 |
| **Type** | Valid |
| **Description** | 在密钥管理模块中存储的密钥，可以通过明文密钥导出接口导出密钥；也可以将明文密钥导入到密钥管理模块中存储； |
| **Upstream ID** | SWS\_CRYPT\_04203 SWS\_CRYPT\_04205 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：密钥管理模块实现完成 ，导入导出接口实现完成  主要测试点：调用密钥类明文导入导出接口，验证结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1 调用密钥类明文导入导出接口，结果正确；  2 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00115]密钥密文导入导出

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00115 |
| **Type** | Valid |
| **Description** | 在密钥管理模块中存储的密钥，可以通过密文密钥导出接口导出密钥；也可以将密文密钥导入到密钥管理模块中存储； |
| **Upstream ID** | SWS\_CRYPT\_04202 SWS\_CRYPT\_04204 SWS\_CRYPT\_04213 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：密钥管理模块实现完成 ，导入导出接口实现完成  主要测试点：调用密钥类密文导入导出接口，验证结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1 调用密钥类密文导入导出接口，结果正确；  2 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00116]密钥类对象加载

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00116 |
| **Type** | Valid |
| **Description** | 在对数据进行加密处理过程中，密钥是不能暴露于处理流程中的，密钥的使用一般都是通过密钥对象加在接口来获取密钥访问信息的。密钥类对象加载可以对 秘密种子，对称密钥，公钥，私钥以及其他加密对象； |
| **Upstream ID** | SWS\_CRYPT\_04200 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：密钥类对象加载接口实现完成  主要测试点：调用密钥类对象加载接口，验证结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1 调用密钥类对象加载接口，结果正确；  2 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00117]申请易失性容器

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00117 |
| **Type** | Valid |
| **Description** | 申请易失性容器，进行密钥导入操作； |
| **Upstream ID** | SWS\_CRYPT\_04208 SWS\_CRYPT\_40959 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：接口实现完成  主要测试点：调用接口，结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1 调用接口，结果正确；  2 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00118]算法ID与名称转换

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00118 |
| **Type** | Valid |
| **Description** | Crypto模块中定义了很多算法，算法的命名要满足标准要求。提供算法ID与名称相互转换接口； |
| **Upstream ID** | SWS\_CRYPT\_40970 SWS\_CRYPT\_40971 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：接口实现完成  主要测试点：调用接口，转换结果是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1 调用接口，结果正确；  2 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00119]获取实例化CryptoProvider

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00119 |
| **Type** | Valid |
| **Description** | Crypto模块支持多CryptoProvider提供加密服务，可根据指定的CryptoProvider信息加载CryptoProvider实例，提供加密服务； |
| **Upstream ID** | SWS\_CRYPT\_00005 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：NeuSAR-aCore平台  前提条件：接口实现完成  主要测试点：调用接口，获取的CryptoProvider是否正确；接口异常测试，返回的错误码是否正确；  成功标准：  1 调用接口，获取CryptoProvider结果正确；  2 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00134] Bridging domains:IOinterface

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-Crypto-00134 |
| **Type** | Valid |
| **Description** | FC Crypto 的一个主要设计决策是尽可能分离处理密码学 (crypto::cryp)、密钥管理 (crypto::keys) 和证书管理 (crypto::x509) 的三个域。 为了简化域和来自实际对象的抽象接口之间的交互，引入了 IOInterface 接口作为持久资源和运行时对象之间的中间层。 IOInterface 代表一个智能包装器，提供对其封装内容的访问和元数据。 |
| **Upstream ID** | SWS\_CRYPT\_40947 SWS\_CRYPT\_40948 SWS\_CRYPT\_40949  SWS\_CRYPT\_40950 SWS\_CRYPT\_40951 SWS\_CRYPT\_40952  SWS\_CRYPT\_40953 SWS\_CRYPT\_40954 SWS\_CRYPT\_40955  SWS\_CRYPT\_40956 SWS\_CRYPT\_40957 User\_defined\_00016 |
| **Dependencies** |  |
| **Verification method** | 评审与测试 |
| **Verification Criteria** | 验证环境：虚拟机环境  前提条件：接口实现完成  主要测试点：实现中是否实现了IOinterface类；该类中的方法是否实现；接口异常测试，返回的错误码是否正确；  成功标准：  1 实现了IOinterface类，  2 该类中的各个方法能够正确实现功能，功能验证成功；  3 接口异常测试，返回的错误码正确； |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **CR** | - |
| **Risk** | 无 |
| **Change Type** | QM(C) |

## Key Storage Provider

Key Storage Provider（KSP，命名空间 ara::crypto::keys）负责对不同类型密钥材料（公共、私有、加密密钥或种子）和其他安全关键加密对象（ 数字签名、哈希、MAC HMAC 标签）进行密钥存储。这些加密对象表示为 KeySlot，并提供给应用程序的密钥存储等相关的API，以实现应用程序对密钥的管理。

### [SWRD-Crypto-00200]Keyslot的应用

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD-Crypto-00200 |
| **Type** | Valid |
| **Description** | 应用程序使用的 KeySlot进行密钥管理的相关操作。KeySlot 由应用程序集成者通过 CryptoKeySlot， CryptoKeySlotInterface 和CryptoKeySlotToPortPrototypeMapping在清单中定义。  1）FC Crypto 为每个 AdaptiveApplicationSwComponentType 提供对 CryptoKeySlot 的访问。在应用程序设计中，每个 CryptoKeySlot 都由 CryptoKeySlotInterface 类型的 RPortPrototype 表示。  2）FC Crypto 将为每个应用进程生成一份加密的清单，清单中描述了将 CryptoKeySlot 分配给 CryptoProvider。因此，通过使用由 CryptoKeySlotInterface 键入的 RPortPrototype，可以建立对 CryptoProvider 的分配。  3）CryptoAPI 应提供获取 CryptoProvider 的功能。通过调用 MyProvider()，能够获取 KeySlot 的相应 **CryptoProvider。** |
| **Upstream ID** | User\_defined\_00024、  SWS\_CRYPT\_10000、  SWS\_CRYPT\_10003、  SWS\_CRYPT\_10005、  TPS\_MANI\_03263、  TPS\_MANI\_03264 |
| **Dependencies** | 无 |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **Verification Criteria** | 验证方法：ST  验证环境：上位机及NeuSAR-aCore运行环境  前提条件：上位机上配置依赖加密算法的进程及相关配置  主要测试点：   1. 上位机上加密CryptoKeySlot相关配置可配，且应用进程可引用CryptoKeySlotInterface。 2. 上位机Generate Code后，应用进程文件夹下可生成加密的manifest。 3. 对于aCore平台内的依赖加密的应用进程，系统运行环境内，可查看应用进程下manifest 对加密配置的引用。   成功标准：  1．满足测试点1和2的上位机配置，以及运行环境满足测试点3。 |
| **CR** |  |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00201]密钥对象的安全存储

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD-Crypto-00201 |
| **Type** | Valid |
| **Description** | FC Crypto 通过KSP确保处理过程中密钥对象和存储的密钥对象的机密性和可靠性。  “密钥管理”功能分为四个部分：  1. Key Storage Provider API（命名空间 crypto::keys）。  2. 完整的Certificate Management Provider API（命名空间crypto::x509）。  3. 密钥材料生成、安全导出、公共/安全导入和辅助 API（通过 crypto::cryp::Crypto Provider 接口的方法）。这些方法代表所有需要实现密钥加密转换的操作。HSM 的使用是在硬件中实现的，因此可能无法像软件解决方案那样支持所有 API。  4. 公共加密对象的通用序列化（通过 crypto::Serializable 接口）。 考虑到第 3 类“密钥管理”子 API 与其他加密功能的深度依赖，可能重写某些功能（包括 HSM 领域中对密钥材料的访问控制机制），将其与 Crypto Provider API 的sub-API分开实现没有意义。  密钥存储和证书管理是通过不同的接口实现，因为它们可以完全独立地实现。 这样也允许将不同供应商提供的两者结合起来使用。  此外，KSP还支持ara::crypto::Serializable派生接口ara::crypto::Serializable::ExportPublicly，按属性导出任何不需要额外保证其完整性或机密性公共对象。 |
| **Upstream ID** | SWS\_CRYPT\_10004、  User\_defined\_00025、  SWS\_CRYPT\_10200 |
| **Dependencies** | 无 |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **Verification Criteria** | 验证方法：ST  验证环境：NeuSAR-aCore运行环境  前提条件：有密钥对象存储到加密数据库中  主要测试点：   1. 通过调Crypto Provider的相关接口生成和存储密钥对象。 2. 通过调Crypto Provider的相关接口利用密钥槽号继续密钥运算处理。   成功标准：   1. 满足测试点1，系统运行环境中已密文的形式存在密钥对象或密钥对象存储到HSM安全芯片中，不可见。 2. 满足测试点2，运算过程密钥对象不可见。 |
| **CR** |  |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00202]支持密钥生成

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD-Crypto-00202 |
| **Type** | Valid |
| **Description** | 密钥生成是生成加密密钥的过程。根据使用的加密算法，有两种类型的密钥生成：  **a. 对称算法**：对称系统由一个密钥组成，该密钥在不同方之间共享。  **b.非对称算法**：非对称系统由生成的公钥和私钥组成。 公钥用于加密、密钥封装或签名验证。 私钥用于解密、密钥封装、密钥交换或数字签名计算。  1. KSP应支持对称密钥算法  FC Crypto 通过调用函数 ara::crypto::cryp::CryptoProvider::Generate-SymmetricKey 分配一个新的对称密钥对象。KSP在此过程中应支持该函数处理密钥生成过程。在处理过程中，该函数返回：   * kUnknownIdentifier 错误，依据KSP对CryptoAlgId限制，判断ara::crypto::CryptoAlgId 为不支持的值。 * kIncompatibleArguments 错误，依据KSP对allowedUsage配置，判断allowedUsage 与 ara::crypto::CryptoAlgId 指定的目标算法不兼容。   2. KSP应支持非对称密钥算法  FC Crypto支持非对称密钥生成， CrypotAPI 提供了这样的功能。通过调用ara::crypto::cryp::CryptoProvider::GeneratePrivateKey生成私钥。并通过调用GetPublicKey从私钥对象中获取公钥。在此过程中KPS，需要配合CryptoProvider 完成一些属性的检查。  同时，由于私钥和公钥是紧密耦合的，它们应该具有相同的 COUID，即COUID 应为私钥和公钥共享ID。 |
| **Upstream ID** | SWS\_CRYPT\_10300、  SWS\_CRYPT\_10301、  SWS\_CRYPT\_10303、  SWS\_CRYPT\_10304、  SWS\_CRYPT\_10305、  SWS\_CRYPT\_10306 |
| **Dependencies** | 无 |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **Verification Criteria** | 验证方法：ST  验证环境：NeuSAR-aCore运行环境  前提条件：有密钥对象存储到加密数据库中  主要测试点：   1. 调Crypto Provider的GenerateSymmetricKey验证KSP的支持情况。 2. 调Crypto Provider的GeneratePrivateKey验证KSP的支持情况。   成功标准：   1. 满足测试点1和2，KSP对于不支持的CryptoAlgId报错kUnknownIdentifier，对于不兼容的情况报kIncompatibleArgu-ments。 2. 满足测试点2，私钥和公钥共享COUID。 |
| **CR** |  |
| **Risk** | 无 |
| **Change Type** |  |

### [SWRD-Crypto-00203]支持密钥协商

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD-Crypto-00203 |
| **Type** | Valid |
| **Description** | 在密钥协商过程中，对称密钥的机密交换很重要，在Crypto Provider中提供“加密信封”或“数字信封”的密钥封装机制。FC Crypto 提供了KeyAgreementPrivateCtx 和 KeyEncapsulatorPublicCtx这两个上下文，来实现了数据封装机制，此外，通过 HashFunctionCtx 和 SignerPrivateCtx 提供的数字签名来确保不可否认性。因此密钥协商的上下文上包含两个构建过程：   * 加密算法 * 解密算法   KSP在密钥协商过程中，主要配合以下处理过程：  ara::crypto::cryp::KeyAgreementPrivateCtx上下文，需要目标密钥协议加密算法的标识符来设置正确的私有上下文；  密钥协议私有上下文提供生成公共秘密种子 ara::crypto::cryp::Secret-Seed的功能；  密钥协议私有上下文应提供生成公共对称密钥的功能。  在以上密钥协商处理过程中，KSP负责配合密钥属性的校验以及密钥的安全导入导出。 |
| **Upstream ID** | SWS\_CRYPT\_10403、  SWS\_CRYPT\_10401、  SWS\_CRYPT\_10402 |
| **Dependencies** | 无 |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **Verification Criteria** | 验证方法：ST  验证环境：NeuSAR-aCore运行环境  前提条件：有密钥对象存储到加密数据库中  主要测试点：   1. 在密钥协商的上下文，验证KSP的支持情况。   成功标准：   1. 满足测试点，KSP对于不支持的CryptoAlgId报错kUnknownIdentifier，对于不兼容的情况报kIncompatibleArgu-ments。 |
| **CR** |  |
| **Risk** | 无 |
| **Change Type** | 新增 |

## X.509 Provider

X.509证书管理提供者（X.509 Provider）负责X.509证书的解析、验证、导出等功能。NeuSAR 的FC Crypto包括单个 X.509 Provider，该Provider的责任是支持公钥基础设施（PKI）的证书和密钥文件的管理，通过相关的API，实现证书及证书链的解析和验证，以及证书的导出等处理。

### [SWRD-Crypto-00300]X.509 Provider的应用

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD-Crypto-00300 |
| **Type** | Valid |
| **Description** | FC Crypto 仅支持 ara::crypto::x509::X509Provider 的单个实例。由于 X.509 Provider 完全独立于 ara::crypto::cryp::CryptoProvider 和 ara::crypto::keys::KeyStorageProvider 实现，不同的供应商可能会提供不同的 X.509 Provider 和 CryptoProvider / KeyStorageProvider。因此，NeuSAR的标准化的 CryptoAPI 保证了这些独立构建块之间的互操作性。 |
| **Upstream ID** | SWS\_CRYPT\_20000 |
| **Dependencies** | 无 |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **Verification Criteria** | 验证方法：ST  验证环境： NeuSAR-aCore运行环境  前提条件：  主要测试点：  1.X.509 Provider提供相对独立的API。  成功标准：  1．X.509的lib可以被单独集成。 |
| **CR** |  |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-Crypto-00301]证书管理

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD-Crypto-00301 |
| **Type** | Valid |
| **Description** | ara::crypto::x509::X509Provider提供根证书路径上所有需要的 CA，以及对相应公私钥，由 ara::crypto::keys::KeyStorageProvider 处理。  X.509 Provider支持导出证书、证书链。但不支持私钥导出。对于错误的证书不会导出，调用者将收到错误的通知。  对于证书对应的私钥访问，CryptoAPI 提供与访问证书目标私钥相关的所有必需的功能（例如签名功能）。 目标私钥对接PKI的私钥处理。  对PKI客户端私钥的访问只能在内部使用，并通过 X.509 Provider 接口间接使用。私钥永远不要离开 FC Crypto 的边界。 |
| **Upstream ID** | SWS\_CRYPT\_20002、  SWS\_CRYPT\_20602、  SWS\_CRYPT\_20614、  SWS\_CRYPT\_20616 |
| **Dependencies** | 无 |
| **ASIL** | QM(C) |
| **Status** | Draft |
| **Priority** | H |
| **Verification Criteria** | 验证方法：ST  验证环境： NeuSAR-aCore运行环境  前提条件：PKI已提供有效的证书和私钥文件  主要测试点：   * + - 1. 通过X.509 Provider提供的API 进行证书或证书链的的解析、导出等操作。   成功标准：  1．能够正确解析证书或证书链，并利用正确的根证书验签成功；能够导出证书或证书链。 |
| **CR** |  |
| **Risk** | 无 |
| **Change Type** | 新增 |

# 非功能需求(Non-Functional Requirements)

## 制约(Constraint)

*[*

1. *软件运行环境：考虑以下几个方面：*

*接口：与其他模块的接口；*

*环境：使用到其他模块数据类型，AutoSAR头文件的包含关系；*

1. *法律、法规和标准：如：必须按照ISO26262标准进行开发。]*

### [SWRD-ID]非功能需求1(Non-Function Requirement No.1)

|  |  |
| --- | --- |
| **SWRD-ID** |  |
| **Type** |  |
| **Description** |  |
| **Upstream ID** |  |
| **Dependencies** |  |
| **Verification method** |  |
| **Verification Criteria** |  |
| **ASIL** |  |
| **Status** |  |
| **Priority** |  |
| **CR** |  |
| **Risk** |  |
| **Change Type** |  |

### [SWRD-ID]非功能需求2(Non-Function Requirement No.2)

|  |  |
| --- | --- |
| **SWRD-ID** |  |
| **Type** |  |
| **Description** |  |
| **Upstream ID** |  |
| **Dependencies** |  |
| **Verification method** |  |
| **Verification Criteria** |  |
| **ASIL** |  |
| **Status** |  |
| **Priority** |  |
| **CR** |  |
| **Risk** |  |
| **Change Type** |  |

## 性能质量要求(Performance Quality Requirements)

*[从可用性，效率，可维护性，可移植性，可扩展性和性能等角度进行分析。*

1. *可用性：使用，操作相关需求*
2. *效率性：技术软件的反应速度和内存使用量，如：对Rom和Ram的使用量的约束、对内存大小、硬盘大小的约束，对CPU的约束等。*

*如：CPU符合率不满50%等，NVM相关使用参考“NVMList”*

1. *可维护性：变更解析相关、操作方式变化、运行环境的变化、接口变化、精度、时间性能等的需求发生变化时，该软件对这些变化的适应能力要求*
2. *可移植性：向其他制品的展开相关要求*
3. *可扩展性：派生开发要求*
4. *性能：软件相关的性能：如：数据精度要求，说明该软件的输入输出的的数据精度要求，可能包含传输过程中的精度要求；*

*时间性能要求，包括：响应时间、更新处理时间，数据转换和传送时间要求等]*

### [SWRD-ID]错误状态机制

|  |  |
| --- | --- |
| **SWRD-ID** |  |
| **Type** |  |
| **Description** | Crypto Stack API 应支持错误状态通知的有效机制;应提供有关检测到的错误状态的全面信息。 该信息应该足以识别错误条件并决定如何从错误状态中恢复并继续执行。 交付机制应方便应用程序的开发人员并满足 Autosar AP C++14 编码指南 |
| **Upstream ID** | RS\_CRYPTO\_02310 |
| **Dependencies** |  |
| **Verification method** | 评审 |
| **Verification Criteria** | 验证环境：无  前提条件：加密模块设计与实现完成；  主要测试点：错误状态返回通知机制是否合理；可否根据错误信息获取到错误点；编码是否满足要求；  成功标准：   1. 错误相关设计机制，可以能够满足需求； 2. 编码格式满足需求； |
| **ASIL** |  |
| **Status** | Draft |
| **Priority** | H |
| **CR** |  |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD-ID]非功能需求4(Non-Function Requirement No.4)

|  |  |
| --- | --- |
| **SWRD-ID** |  |
| **Type** |  |
| **Description** |  |
| **Upstream ID** |  |
| **Dependencies** |  |
| **Verification method** |  |
| **Verification Criteria** |  |
| **ASIL** |  |
| **Status** |  |
| **Priority** |  |
| **CR** |  |
| **Risk** |  |
| **Change Type** |  |

# 接口说明(API)

*[记录AutoSar中接口相关信息，包括：API，服务接口，错误码等。该章节内容也可以引用“软件接口设计书”，在此处引用该文件即可。]*

## 接口头文件(API Header files)

加密头文件位于ara/crypto下。

## 接口共同数据类型(API Common Data Types)

### [SWRD-API-Crypto-01001] AllowedUsageFlags

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-01001 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10015 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | type alias |
| **Symbol** | AllowedUsageFlags |
| **Scope** | namespace ara::crypto |
| **Derived from** | std::uint32\_t |
| **Syntax** | using AllowedUsageFlags = std::uint32\_t; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 允许使用密钥或秘密种子对象的容器类型和常量位标志。只允许直接指定密钥的使用，其它都被禁止！为原始密钥/种子的使用限制以及可能从原始密钥/种子派生的对称密钥或种子定义了类似的标志集。只有当它支持 kAllowKeyAgreement 或 kAllowKeyDiversify 或 kAllowKeyDerivation 时，才能从原始密钥或秘密种子派生对称密钥或秘密种子！ |

### [SWRD-API-Crypto-01002] ByteVector

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-01002 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10042 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| **Kind** | type alias | |
| **Symbol** | ByteVector | |
| **Scope** | namespace ara::crypto | |
| **Derived from** | ara::core::Vector<std::uint8\_t, Alloc> | |
| **Syntax** | using ByteVector = ara::core::Vector<std::uint8\_t, Alloc>; | |
| **Template param** | Alloc | 字节序列的自定义alloc |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" | |
| **Description** | 带有可自定义分配器的字节向量模板的别名。 | |

### [SWRD-API-Crypto-01003] CryptoAlgId

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-01003 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10014 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | type alias |
| **Symbol** | CryptoAlgId |
| **Scope** | namespace ara::crypto |
| **Derived from** | std::uint64\_t |
| **Syntax** | using CryptoAlgId = std::uint64\_t; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 加密算法标识符的容器类型。 |

### [SWRD-API-Crypto-01004] CryptoObjectType

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-01004 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10016 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| **Kind** | enumeration | |
| **Symbol** | CryptoObjectType | |
| **Scope** | namespace ara::crypto | |
| **Underlying type** | std::uint32\_t | |
| **Syntax** | enum class CryptoObjectType : std::uint32\_t {...}; | |
| **Values** | kUndefined= 0 | 当前未定义（空的container） |
| kSymmetricKey= 1 | cryp::SymmetricKey 对象 |
| kPrivateKey= 2 | cryp::PrivateKey 对象 |
| kPublicKey= 3 | cryp::PublicKey 对象 |
| kSignature= 4 | crypt::Signature 对象（非对称数字签名  或对称 MAC/HMAC 或哈希摘要） |
| kSecretSeed= 5 | cryp::SecretSeed对象  Note: the seed cannot have an associated crypto algorithm!  注意：种子不能有一个相关的加密算法！ |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" | |
| **Description** | 枚举所有类型的加密对象，即可以存储到密钥槽的内容类型。 | |

### [SWRD-API-Crypto-01005] CryptoObjectUid

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-01005 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10100 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | struct |
| **Symbol** | CryptoObjectUid |
| **Scope** | namespace ara::crypto |
| **Syntax** | struct CryptoObjectUid {...}; |
| **Header file** | #include "ara/crypto/common/crypto\_object\_uid.h" |
| **Description** | 加密对象唯一标识符 (COUID) 类型的定义。 |

### [SWRD-API-Crypto-01006] ProviderType

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-01006 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10017 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| **Kind** | enumeration | |
| **Symbol** | ProviderType | |
| **Scope** | namespace ara::crypto | |
| **Underlying type** | std::uint32\_t | |
| **Syntax** | enum class ProviderType : std::uint32\_t {...}; | |
| **Values** | kUndefinedProvider= 0 | 未定义/未知的Provider类型（或适用于全部的Crypto Stack） |
| kCryptoProvider= 1 | Cryptography Provider. |
| kKeyStorageProvider= 2 | Key Storage Provider. |
| kX509Provider= 3 | X.509 Provider. |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" | |
| **Description** | 枚举所有已知的 Provider 类型。 | |

### [SWRD-API-Crypto-01007] ReadOnlyMemRegion

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-01007 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10033 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | type alias |
| **Symbol** | ReadOnlyMemRegion |
| **Scope** | namespace ara::crypto |
| **Derived from** | ara::core::Span<const std::uint8\_t> |
| **Syntax** | using ReadOnlyMemRegion = ara::core::Span<const std::uint8\_t>; |
| **Header file** | #include "ara/crypto/common/mem\_region.h" |
| **Description** | 只读内存区域（用于 [in] 参数） |

### [SWRD-API-Crypto-01008]ReadWriteMemRegion

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-01008 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10031 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | type alias |
| **Symbol** | ReadWriteMemRegion |
| **Scope** | namespace ara::crypto |
| **Derived from** | ara::core::Span<const std::uint8\_t> |
| **Syntax** | using ReadWriteMemRegion = ara::core::Span<std::uint8\_t>; |
| **Header file** | #include "ara/crypto/common/mem\_region.h" |
| **Description** | 读写内存区域（用于 [in/out] 参数） |

### [SWRD-API-Crypto-01009] CryptoErrc

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-01009 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10099 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| **Kind** | enumeration | |
| **Symbol** | CryptoErrc | |
| **Scope** | namespace ara::crypto | |
| **Underlying type** | ara::core::ErrorDomain::CodeType | |
| **Syntax** | enum class CryptoErrc : ara::core::ErrorDomain::CodeType {...}; | |
| **Values** | kErrorClass= 0x1000000 | 保留（错误类别 ID 的乘数） |
| kErrorSubClass= 0x10000 | 保留（错误子类 ID 的乘数） |
| kErrorSubSubClass= 0x100 | 保留（错误子类 ID 的乘数） |
| kResourceFault=1 \* kErrorClass | ResourceException：Generic resource fault! |
| kBusyResource=kResourceFault + 1 | ResourceException：Specified resource is busy! |
| kInsufficientResource=kResourceFault  + 2 | ResourceException:Insufficient capacity of specified resource! |
| kUnreservedResource=kResourceFault + 3 | ResourceException: Specified resource was not  reserved! |
| kModifiedResource=kResourceFault +4 | ResourceException: Specified resource has been  modified! |
| kLogicFault= 2 \* kErrorClass | LogicException: Generic logic fault! |
| kInvalidArgument= kLogicFault + 1 \* k  ErrorSubClass | InvalidArgumentException:An invalid argument value is provided! |
| kUnknownIdentifier=kInvalidArgument  + 1 | InvalidArgumentException: Unknown identifier is provided! |
| kInsufficientCapacity=InvalidArgument + 2 | InvalidArgumentException: Insufficient capacity of the output buffer! |
| kInvalidInputSize= kInvalidArgument +  3 | InvalidArgumentException: Invalid size of an input buffer! |
| kIncompatibleArguments= kInvalid  Argument + 4 | InvalidArgumentException: Provided values of arguments are incompatible! |
| kInOutBuffersIntersect= kInvalid  Argument + 5 | InvalidArgumentException: Input and output buffers are intersect! |
| kBelowBoundary= kInvalidArgument +  6 | InvalidArgumentException: Provided value is below the lower boundary! |
| kAboveBoundary= kInvalidArgument +  7 | InvalidArgumentException: Provided value is above the upper boundary! |
| kAuthTagNotValid= kInvalidArgument + 8 | AuthTagNotValidException: Provided  authentication-tag cannot be verified! |
| kUnsupported= kInvalidArgument + 1 \*  kErrorSubSubClass | UnsupportedException: Unsupported request (due  to limitations of the implementation)! |
| kInvalidUsageOrder= kLogicFault + 2\*  kErrorSubClass | InvalidUsageOrderException: Invalid usage order of the interface! |
| kUninitializedContext= kInvalidUsage  Order + 1 | InvalidUsageOrderException: Context of the interface was not initialized! |
| kProcessingNotStarted= kInvalidUsage  Order + 2 | InvalidUsageOrderException: Data processing was not started yet! |
| kProcessingNotFinished= kInvalid  UsageOrder + 3 | InvalidUsageOrderException: Data processing was not finished yet! |
| kRuntimeFault= 3 \* kErrorClass | RuntimeException: Generic runtime fault! |
| kUnsupportedFormat= kRuntimeFault  + 1 | RuntimeException: Unsupported serialization  format for this object type! |
| kBruteForceRisk= kRuntimeFault + 2 | RuntimeException: Operation is prohibitted due to a  risk of a brute force attack! |
| kContentRestrictions=kRuntimeFault+3 | RuntimeException: The operation violates content restrictions of the target container! |
| kBadObjectReference=kRuntimeFault+ 4 | RuntimeException: Incorrect reference between objects! |
| kContentDuplication=kRuntimeFault +  6 | RuntimeException:Provided content already exists in the target storage! |
| kUnexpectedValue= kRuntimeFault + 1  \* kErrorSubClass | UnexpectedValueException: Unexpected value of an argument is provided! |
| kIncompatibleObject= kUnexpected  Value + 1 | UnexpectedValueException: The provided object is incompatible with requested operation or its configuration! |
| kIncompleteArgState= kUnexpected  Value + 2 | UnexpectedValueException: Incomplete state of an argument! |
| kEmptyContainer= kUnexpectedValue  + 3 | UnexpectedValueException: Specified container is empty! |
| kMissingArgument=UnexpectedValue  + 4 | kMissingArgumentException: Expected argument, but none provided! |
| kBadObjectType= kUnexpectedValue +  1 \* kErrorSubSubClass | BadObjectTypeException: Provided object has unexpected type! |
| kUsageViolation= kRuntimeFault + 2 \*  kErrorSubClass | UsageViolationException: Violation of allowed usage for the object! |
| kAccessViolation= kRuntimeFault + 3 \*  kErrorSubClass | AccessViolationException: Access rights violation! |
| **Header file** | #include "ara/crypto/common/crypto\_error\_domain.h" | |
| **Description** | ara::crypto 可能报告的所有加密错误代码值的枚举。 | |

### [SWRD-API-Crypto-01010] SecureCounter

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-01010 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30001 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | struct |
| **Symbol** | SecureCounter |
| **Scope** | namespace ara::crypto |
| **Syntax** | struct SecureCounter {...}; |
| **Header file** | #include "ara/crypto/common/entry\_point.h" |
| **Description** | 由硬件计数器的最高有效和最低有效四字组成的 128 位安全计数器。 |

### [SWRD-API-Crypto-01011] FormatId

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-01011 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10701 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | type alias |
| **Symbol** | FormatId |
| **Scope** | class ara::crypto::Serializable |
| **Derived from** | std::uint32\_t |
| **Syntax** | using FormatId = std::uint32\_t; |
| **Header file** | #include "ara/crypto/common/serializable.h" |
| **Description** | 编码格式标识符的容器类型。 |

### [SWRD-API-Crypto-01012] CryptoTransform

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-01012 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10019 | |
| **CR** |  | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| **Kind** | enumeration | |
| **Symbol** | CryptoTransform | |
| **Scope** | namespace ara::crypto | |
| **Underlying type** | std::uint32\_t | |
| **Syntax** | enum class CryptoTransform : std::uint32\_t {...}; | |
| **Values** | kEncrypt= 1 | encryption |
| kDecrypt= 2 | decryption |
| kMacVerify= 3 | MAC verification. |
| kMacGenerate= 4 | MAC generation. |
| kWrap= 5 | key wrapping |
| kUnwrap= 6 | key unwrapping |
|  | kSigVerify= 7 | signature verification |
|  | kSigGenerate= 8 | signature generation |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" | |
| **Description** | 加密转换的枚举。 | |

### [SWRD-API-Crypto-01013] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-01013 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10852 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::VolatileTrustedContainer |
| **Derived from** | std::unique\_ptr<VolatileTrustedContainer> |
| **Syntax** | using Uptr = std::unique\_ptr<VolatileTrustedContainer>; |
| **Header file** | #include "ara/crypto/common/volatile\_trusted\_container.h" |
| **Description** | 接口的唯一智能指针。 |

### [SWRD-API-Crypto-01014] Uuid

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-01014 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10400 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | struct |
| **Symbol** | Uuid |
| **Scope** | namespace ara::crypto |
| **Syntax** | struct Uuid {...}; |
| **Header file** | #include "ara/crypto/common/uuid.h" |
| **Description** | Definition of Universally Unique Identifier (UUID) type. Independently from internal definition details of this structure, it’s size must be 16 bytes and entropy of this ID should be close to 128 bit!  通用唯一标识符 (UUID) 类型的定义。 独立于这个结构的内部定义细节，它的大小必须是 16 字节，这个 ID 的熵应该接近 128 位！ |

### [SWRD-API-Crypto-01015] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-01015 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10801 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::IOInterface |
| **Derived from** | std::unique\_ptr<IOInterface> |
| **Syntax** | using Uptr = std::unique\_ptr<IOInterface>; |
| **Header file** | #include "ara/crypto/common/io\_interface.h" |
| **Description** | 接口的唯一智能指针。 |

### [SWRD-API-Crypto-01016] Uptrc

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-01016 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10802 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | type alias |
| **Symbol** | Uptrc |
| **Scope** | class ara::crypto::IOInterface |
| **Derived from** | std::unique\_ptr<const IOInterface> |
| **Syntax** | using Uptrc = std::unique\_ptr<const IOInterface>; |
| **Header file** | #include "ara/crypto/common/io\_interface.h" |
| **Description** | 常量接口的唯一智能指针。 |

### [SWRD-API-Crypto-01017] Errc

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-01017 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_19903 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | type alias |
| **Symbol** | Errc |
| **Scope** | class ara::crypto::CryptoErrorDomain |
| **Derived from** | CryptoErrc |
| **Syntax** | using Errc = CryptoErrc; |
| **Header file** | #include "ara/crypto/common/crypto\_error\_domain.h" |
| **Description** | crypto error |

## 接口定义(API Reference)

#### [SWRD-API-Crypto-00001]AuthCipherCtx

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00001 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20100 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | AuthCipherCtx |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoContext |
| **Syntax** | class AuthCipherCtx : public CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/auth\_cipher\_ctx.h" |
| **Description** | 认证加密上下文接口。 派生接口 BufferedDigest 的方法用于对关联的公共数据进行身份验证。 派生接口 StreamCipherCtx 的方法用于消息的机密部分的加密/解密和认证。 数据处理必须按以下顺序执行：  1 调用 Start() 方法之一。  2 通过调用 Update() 方法处理所有相关的公共数据。  3 通过调用 ProcessBlocks()、Process Bytes()（以及可选的 FinishBytes()）方法来处理消息的机密部分。  4 调用 Finish() 方法以完成验证码计算（并可选择获取）。  5 可以提取计算出的 MAC 的副本（通过 GetDigest()）或在内部进行比较（通过 Compare()）。 在整个解密和认证过程完成之前，接收方不应使用解密的数据！ 例如，解密后的数据只有在MAC验证成功后才能使用！  注意：上述描述部分取自AutoSAR文档，但是描述中提到的类或者方法是在头文件以及标准找不到的。所以此部分内容标准中存在问题； |

#### [SWRD-API-Crypto-00002]BlockService

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00002 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_29030 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | BlockService |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | ExtensionService |
| **Syntax** | class BlockService : public ExtensionService {...}; |
| **Header file** | #include "ara/crypto/cryp/block\_service.h" |
| **Description** | 块加密上下文的扩展元信息服务。 |

#### [SWRD-API-Crypto-00003] CryptoContext

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00003 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20400 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | CryptoContext |
| **Scope** | namespace ara::crypto::cryp |
| **Syntax** | class CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/crypto\_context.h" |
| **Description** | 可变加密上下文的通用接口，即未绑定到单个加密对象。 |

#### [SWRD-API-Crypto-00004] CryptoObject

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00004 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20500 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | CryptoObject |
| **Scope** | namespace ara::crypto::cryp |
| **Syntax** | class CryptoObject {...}; |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_object.h" |
| **Description** | CryptoProvider 可识别的所有密码对象的通用接口。 此接口（或其任何派生类）表示可加载到临时转换上下文的非可变（完成后）对象 |

#### [SWRD-API-Crypto-00005] CryptoPrimitiveId

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00005 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20600 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | CryptoPrimitiveId |
| **Scope** | namespace ara::crypto::cryp |
| **Syntax** | class CryptoPrimitiveId{...}; |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_primitive\_id.h" |
| **Description** | 用于识别所有加密算法及其密钥和参数的通用接口 |

#### [SWRD-API-Crypto-00006] CryptoProvider

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00006 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20700 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | CryptoProvider |
| **Scope** | namespace ara::crypto::cryp |
| **Syntax** | class CryptoProvider {...}; |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" |
| **Description** | Crypto Provider是所有支持的Crypto Primitives的“工厂”接口，以及它们之间的内部通信的“可信环境”。所有Crypto Primitives都应该有一个对其父Crypto Provider的实际引用。Crypto Provider只有在销毁其所有子代Crypto Primitives后才能被销毁。该接口中创建Crypto Primitive实例的每个方法都是非常量，因为任何这样的创建都会增加Crypto Primitive的引用计数器。 |

*.*

#### [SWRD-API-Crypto-00007] CryptoService

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00007 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_29020 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | CryptoService |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | ExtensionService |
| **Syntax** | class CryptoService : public ExtensionService {...}; |
| **Header file** | #include "ara/crypto/cryp/crypto\_service.h" |
| **Description** | 加密上下文的扩展元信息服务 |

#### [SWRD-API-Crypto-00008] DecryptorPrivateCtx

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00008 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20800 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | DecryptorPrivateCtx |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoContext |
| **Syntax** | class DecryptorPrivateCtx : public CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/decryptor\_private\_ctx.h" |
| **Description** | 非对称解密私钥上下文接口。 |

#### [SWRD-API-Crypto-00009] DigestService

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00009 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_29010 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | DigestService |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | BlockService |
| **Syntax** | class DigestService : public BlockService {...}; |
| **Header file** | #include "ara/crypto/cryp/digest\_service.h" |
| **Description** | 摘要生成上下文的扩展元信息服务。 |

#### [SWRD-API-Crypto-00010] EncryptorPublicCtx

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00010 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_21000 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | EncryptorPublicCtx |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoContext |
| **Syntax** | class EncryptorPublicCtx : public CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/encryptor\_public\_ctx.h" |
| **Description** | 非对称加密公钥上下文接口。 |

#### [SWRD-API-Crypto-00011] ExtensionService

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00011 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_29040 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | ExtensionService |
| **Scope** | namespace ara::crypto::cryp |
| **Syntax** | class ExtensionService {...}; |
| **Header file** | #include "ara/crypto/cryp/extension\_service.h" |
| **Description** | 适用于所有上下文的基本元信息服务。 |

#### [SWRD-API-Crypto-00012] HashFunctionCtx

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00012 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_21100 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | HashFunctionCtx |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoContext |
| **Syntax** | class HashFunctionCtx : public CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/hash\_function\_ctx.h" |
| **Description** | 散列功能接口。 |

#### [SWRD-API-Crypto-00013] KeyAgreementPrivateCtx

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00013 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_21300 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | KeyAgreementPrivateCtx |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoContext |
| **Syntax** | class KeyAgreementPrivateCtx : public CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/key\_agreement\_private\_ctx.h" |
| **Description** | 密钥协议私钥上下文接口（Diffie Hellman 或概念上类似）。 |

#### [SWRD-API-Crypto-00014] KeyDecapsulatorPrivateCtx

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00014 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_21400 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | KeyDecapsulatorPrivateCtx |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoContext |
| **Syntax** | class KeyDecapsulatorPrivateCtx : public CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/key\_decapsulator\_private\_ctx.h" |
| **Description** | 非对称密钥封装机制 (KEM) 私钥上下文接口。 |

#### [SWRD-API-Crypto-00015] KeyDerivationFunctionCtx

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00015 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_21500 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | KeyDerivationFunctionCtx |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoContext |
| **Syntax** | class KeyDerivationFunctionCtx : public CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/key\_derivation\_function\_ctx.h" |
| **Description** | 密钥派生功能接口。 |

#### [SWRD-API-Crypto-00016] KeyEncapsulatorPublicCtx

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00016 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_21800 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | KeyEncapsulatorPublicCtx |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoContext |
| **Syntax** | class KeyEncapsulatorPublicCtx : public CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/key\_encapsulator\_public\_ctx.h" |
| **Description** | 称密钥封装机制 (KEM) 公钥上下文接口。 |

#### [SWRD-API-Crypto-00017 MessageAuthnCodeCtx

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00017 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_22100 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | MessageAuthnCodeCtx |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoContext |
| **Syntax** | class MessageAuthnCodeCtx : public CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/message\_authn\_code\_ctx.h" |
| **Description** | 钥消息认证代码上下文接口定义 (MAC/HMAC)。 |

#### [SWRD-API-Crypto-00018] MsgRecoveryPublicCtx

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00018 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_22200 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | MsgRecoveryPublicCtx |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoContext |
| **Syntax** | class MsgRecoveryPublicCtx : public CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/msg\_recovery\_public\_ctx.h" |
| **Description** | 用于非对称恢复短消息及其签名验证（类似 RSA）的公钥上下文。 如果合理的生成公钥并且对其保密，受限制的可信订阅户组可以使用此原语用于提供短消息的机密性、真实性和不可否认性。 |

#### [SWRD-API-Crypto-00019] PrivateKey

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00019 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_22500 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | PrivateKey |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | RestrictedUseObject |
| **Syntax** | class PrivateKey : public RestrictedUseObject {...}; |
| **Header file** | #include "ara/crypto/cryp/cryobj/private\_key.h" |
| **Description** | 非对称私钥接口 |

#### [SWRD-API-Crypto-00020] PublicKey

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00020 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_22700 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | PublicKey |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | RestrictedUseObject |
| **Syntax** | class PublicKey : public RestrictedUseObject {...}; |
| **Header file** | #include "ara/crypto/cryp/cryobj/public\_key.h" |
| **Description** | 非对称公钥接口 |

#### [SWRD-API-Crypto-00021] RandomGeneratorCtx

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00021 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_22900 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | RandomGeneratorCtx |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoContext |
| **Syntax** | class RandomGeneratorCtx : public CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/random\_generator\_ctx.h" |
| **Description** | 随机数生成上下文接口 |

#### [SWRD-API-Crypto-00022] RestrictedUseObject

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00022 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_24800 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | RestrictedUseObject |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoObject |
| **Syntax** | class RestrictedUseObject : public CryptoObject {...}; |
| **Header file** | #include "ara/crypto/cryp/cryobj/restricted\_use\_object.h" |
| **Description** | 支持使用限制的所有对象的通用接口 |

#### [SWRD-API-Crypto-00023] SecretSeed

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00020 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23000 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | SecretSeed |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | RestrictedUseObject |
| **Syntax** | class SecretSeed : public RestrictedUseObject {...}; |
| **Header file** | #include "ara/crypto/cryp/cryobj/secret\_seed.h" |
| **Description** | 秘密种子对象接口。 |

#### [SWRD-API-Crypto-00024] SigEncodePrivateCtx

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00024 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23200 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | SigEncodePrivateCtx |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoContext |
| **Syntax** | class SigEncodePrivateCtx : public CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/sig\_encode\_private\_ctx.h" |
| **Description** | 用于非对称签名计算和短消息编码（类似 RSA）的私钥上下文。如果合理的生成公钥并且对其保密，受限制的可信订阅户组可以使用此原语用于提供短消息的机密性、真实性和不可否认性。 |

#### [SWRD-API-Crypto-00025] SignatureService

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00025 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_29000 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | SignatureService |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | ExtensionService |
| **Syntax** | class SignatureService : public ExtensionService {...}; |
| **Header file** | #include "ara/crypto/cryp/signature\_service.h" |
| **Description** | 签名上下文的扩展元信息服务。 |

#### [SWRD-API-Crypto-00026] Signature

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00026 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23300 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | Signature |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoObject |
| **Syntax** | class Signature : public CryptoObject {...}; |
| **Header file** | #include "ara/crypto/cryp/cryobj/signature.h" |
| **Description** | 签名容器接口用于保存数字签名、哈希摘要、(Hash-based) Message Authentication Code (MAC/HMAC)。在密钥签名(数字签名或MAC/HMAC)的情况下，签名验证密钥的COUID可以通过调用CryptoObject:: hasdependency()获得!。 |

#### [SWRD-API-Crypto-00027] SignerPrivateCtx

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00027 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23500 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | SignerPrivateCtx |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoContext |
| **Syntax** | class SignerPrivateCtx : public CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/signer\_private\_ctx.h" |
| **Description** | 签名私钥上下文接口。 |

#### [SWRD-API-Crypto-00028] StreamCipherCtx

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00028 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23600 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | StreamCipherCtx |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoContext |
| **Syntax** | class StreamCipherCtx : public CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" |
| **Description** | 通用流加密上下文接口（它涵盖了所有操作模式）。 |

#### [SWRD-API-Crypto-00029] SymmetricBlockCipherCtx

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00029 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23700 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | SymmetricBlockCipherCtx |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoContext |
| **Syntax** | class SymmetricBlockCipherCtx : public CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/symmetric\_block\_cipher\_ctx.h" |
| **Description** | 带有填充的对称块密码上下文的接口。 |

#### [SWRD-API-Crypto-00030] SymmetricKey

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00030 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23800 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | SymmetricKey |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | RestrictedUseObject |
| **Syntax** | class SymmetricKey : public RestrictedUseObject {...}; |
| **Header file** | #include "ara/crypto/cryp/cryobj/symmetric\_key.h" |
| **Description** | 对称密钥接口。 |

#### [SWRD-API-Crypto-00031] SymmetricKeyWrapperCtx

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00031 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_24000 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | SymmetricKeyWrapperCtx |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoContext |
| **Syntax** | class SymmetricKeyWrapperCtx : public CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/symmetric\_key\_wrapper\_ctx.h" |
| **Description** | 对称密钥封装算法的上下文(对于AES，它应该兼容RFC3394或RFC5649)。该上下文的公共接口专门用于原始密钥材料的包装/解包装，即不向源加密对象中的密钥材料分配任何元信息。但是除此之外，这种上下文类型应该支持一些适合于整个加密对象导出/导入的“隐藏”低级方法。整个加密对象(包括相关的元信息)的密钥封装可以通过ExportSecuredObject()和ImportSecuredObject()方法完成，但不符合RFC3394或RFC5649。 |

#### [SWRD-API-Crypto-00032] VerifierPublicCtx

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00032 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_24100 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | class |
| **Symbol** | VerifierPublicCtx |
| **Scope** | namespace ara::crypto::cryp |
| **Base class** | CryptoContext |
| **Syntax** | class VerifierPublicCtx : public CryptoContext {...}; |
| **Header file** | #include "ara/crypto/cryp/verifier\_public\_ctx.h" |
| **Description** | 签名验证公钥上下文接口 |

#### [SWRD-API-Crypto-00033] Check

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00033 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20319 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Check(const Signature &expected) | |
| **Scope** | class ara::crypto::cryp::AuthCipherCtx | |
| **Syntax** | virtual ara::core::Result<bool> Check (const Signature &expected) const noexcept=0; | |
| **Parameters (in)** | expected | 包含预期摘要值的签名对象 |
| **Return value:** | ara::core::Result< bool > | 如果提供的“签名”对象的值和元信息分别与计算的摘要和上下文的当前配置相同，则为 true； 但否则为false |
| **Exception Safety:** | noexcept | |
| **Thread Safety:** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Finished | 如果未通过调用 Finish() 方法完成摘要计算 |
| CryptoErrorDomain::kIncompatible Object | 如果提供的“签名”对象是由另一个加密算法类型产生的 |
| **Header file** | #include "ara/crypto/cryp/auth\_cipher\_ctx.h" | |
| **Description** | 根据预期的“签名”对象检查计算的摘要。 整个摘要值保存在上下文中，直到下次调用 Start()，因此可以再次验证或提取它。 此方法可以在函数 ara::core::memcmp() 标准化后实现为“内联” | |
| **Additional** |  | |

#### [SWRD-API-Crypto-00034] GetDigestService

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00034 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20102 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetDigestService() | |
| **Scope** | class ara::crypto::cryp::AuthCipherCtx | |
| **Syntax** | virtual DigestService::Uptr GetDigestService () const noexcept=0; | |
| **Return value** | DigestService::Uptr | - |
| **Exception Safety** | noexcept | |
| **Header file** | #include "ara/crypto/cryp/auth\_cipher\_ctx.h" | |
| **Description** | 获取摘要服务实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00035] GetDigest

|  |  |  |  |
| --- | --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00035 | | |
| **Type** | Valid | | |
| **Priority** | H | | |
| **Upstream ID** | SWS\_CRYPT\_20316 SWS\_CRYPT\_01811 | | |
| **CR** | - | | |
| **Consistency** | Yes | | |
| **Change Type** | 新增 | | |
| ***Kind*** | function | | |
| **Symbol** | GetDigest(std::size\_t offset=0) | | |
| **Scope** | class ara::crypto::cryp::AuthCipherCtx | | |
| **Syntax** | template <typename Alloc = <implementation-defined>> ara::core::Result<ByteVector<Alloc> > GetDigest (std::size\_t offset=0) const noexcept; | | |
| **Parameters (in)** | offset | | 应该放置到输出缓冲区的摘要的第一个字节的位置 |
| **Return value** | ara::core::Result<ByteVector<Alloc> > | | 存储请求的摘要片段或完整摘要的输出缓冲区 |
| **Exception Safety** | noexcept | | |
| **Thread Safety** | Thread-safe | | |
| **Errors** | CryptoErrorDomain::kProcessingNot Finished | 如果未通过调用 Finish() 方法完成摘要计算 | |
| CryptoErrorDomain::kUsageViolation | 如果缓冲的摘要属于由没有 kAllowSignature 许可的密钥初始化的 MAC/HMAC/AE/AEAD 上下文 | |
| **Header file** | #include "ara/crypto/cryp/auth\_cipher\_ctx.h" | | |
| **Description** | 检索计算的摘要。 整个摘要值保留在上下文中，直到下一次调用 Start()。 因此，可以随时重新检查或提取摘要。 如果偏移量大于摘要，则应返回一个空缓冲区。 此方法可以在函数 ara::core::memcpy() 标准化后实现为“内联”。 | | |
| **Additional** | - | | |

#### [SWRD-API-Crypto-00036] GetTransformation

|  |  |  |  |
| --- | --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00036 | | |
| **Type** | Valid | | |
| **Priority** | H | | |
| **Upstream ID** | SWS\_CRYPT\_21715 | | |
| **CR** | - | | |
| **Consistency** | Yes | | |
| **Change Type** | 新增 | | |
| ***Kind*** | function | | |
| **Symbol** | GetTransformation() | | |
| **Scope** | class ara::crypto::cryp::AuthCipherCtx | | |
| **Syntax** | virtual ara::core::Result<CryptoTransform> GetTransformation () const noexcept=0; | | |
| **Return value** | ara::core::Result< CryptoTransform > | 转换方式 | |
| **Exception Safety** | noexcept | | |
| **Thread Safety** | Thread-safe | | |
| **Errors** | CryptoErrorDomain::kUninitialized Context | | 如果在初始化期间可以配置此上下文的转换方向，但尚未初始化上下文 |
| **Header file** | #include "ara/crypto/cryp/auth\_cipher\_ctx.h" | | |
| **Description** | 获取上下文配置的转换方式：加密（kEncrypt）或者解密（kDecrypt） | | |
| **Additional** | - | | |

#### [SWRD-API-Crypto-00037] GetMaxAssociatedDataSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00037 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20103 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetMaxAssociatedDataSize() | |
| **Scope** | class ara::crypto::cryp::AuthCipherCtx | |
| **Syntax** | virtual std::uint64\_t GetMaxAssociatedDataSize () const noexcept=0; | |
| **Return value** | std::uint64\_t | 关联公共数据的最大支持大小（以字节为单位） |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/auth\_cipher\_ctx.h" | |
| **Description** | 获取关联公共数据的最大支持大小。 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-00038] ProcessConfidentialData

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00038 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23634 SWS\_CRYPT\_01800 SWS\_CRYPT\_01803  SWS\_CRYPT\_01804 SWS\_CRYPT\_01805 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ProcessConfidentialData(ReadOnlyMemRegion in, ara::core::Optional< ReadOnlyMemRegion > expectedTag) | |
| **Scope** | class ara::crypto::cryp::AuthCipherCtx | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > Process ConfidentialData (ReadOnlyMemRegion in, ara::core::Optional< ReadOnly MemRegion > expectedTag) noexcept=0; | |
| **Parameters (in)** | in | 包含完整消息的输入缓冲区 |
| expectedTag | 可选指针，指向只读内存区域，其中包含用于验证的认证标签 |
| **Return value** | ara::core::Result<ara::core::Vector<ara::core::Byte> > | 处理后的数据 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidInputSize | 如果输入缓冲区的大小不能被块大小整除（请参阅 GetBlockSize()） |
| CryptoErrorDomain::kProcessingNot Started | 如果数据处理不是通过调用 Start() 方法启动的 |
| CryptoErrorDomain::kAuthTagNotValid | 如果处理后的数据不能被认证 |
| **Header file** | #include "ara/crypto/cryp/auth\_cipher\_ctx.h" | |
| **Description** | 处理机密数据并返回结果。 输入缓冲区将被处理后的消息覆盖。 此函数是最终调用，即必须已经提供了所有相关数据。 因此，该函数将检查身份验证标签，如果标签有效，则仅返回处理后的数据。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00039] ProcessConfidentialData

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00039 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23635 SWS\_CRYPT\_01800 SWS\_CRYPT\_01803  SWS\_CRYPT\_01804 SWS\_CRYPT\_01805 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ProcessConfidentialData(ReadWriteMemRegion inOut, ara::core::Optional< ReadOnlyMem Region > expectedTag) | |
| **Scope** | class ara::crypto::cryp::AuthCipherCtx | |
| **Syntax** | virtual ara::core::Result<void> ProcessConfidentialData (ReadWriteMemRegion inOut, ara::core::Optional< ReadOnlyMemRegion > expectedTag) noexcept=0; | |
| **Parameters (in)** | inOut | 包含完整消息的输入缓冲区 |
| expectedTag | 可选指针，指向只读内存区域，其中包含用于验证的认证标签 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidInputSize | 如果输入缓冲区的大小不能被块大小整除（请参阅 GetBlockSize()） |
| CryptoErrorDomain::kProcessingNot Started | 如果数据处理不是通过调用 Start() 方法启动的 |
| CryptoErrorDomain::kAuthTagNotValid | 如果处理后的数据不能被认证 |
| **Header file** | #include "ara/crypto/cryp/auth\_cipher\_ctx.h" | |
| **Description** | 处理机密数据并使用处理后的消息更新输入缓冲区。 输入缓冲区将被处理后的消息覆盖。调用此方法后，可能不会更新其他关联数据。 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-00040] Reset

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00040 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20414 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Reset() | |
| **Scope** | class ara::crypto::cryp::AuthCipherCtx | |
| **Syntax** | virtual ara::core::Result<void> Reset () noexcept=0; | |
| **Return value** | ara::core::Result<void> | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/auth\_cipher\_ctx.h" | |
| **Description** | 清除加密上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00041] SetKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00041 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23911 SWS\_CRYPT\_01807 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SetKey(const SymmetricKey &key, CryptoTransform transform=CryptoTransform::kEncrypt) | |
| **Scope** | class ara::crypto::cryp::AuthCipherCtx | |
| **Syntax** | virtual ara::core::Result<void> SetKey (const SymmetricKey &key, CryptoTransform transform=CryptoTransform::kEncrypt) noexcept=0; | |
| **Parameters (in)** | key | 对称密钥对象 |
|  | transform | “方向”指示符：部署密钥用于加密（如果为true）或用于解密（如果为false） |
| **Return value** | ara::core::Result<void> | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncompatible Object | 如果提供的密钥对象与此对称密钥上下文不兼容 |
| CryptoErrorDomain::kUsageViolation | 如果与此上下文关联的转换类型（考虑到由转换指定的方向）被提供的密钥对象的“允许使用”限制禁止 |
| **Header file** | #include "ara/crypto/cryp/auth\_cipher\_ctx.h" | |
| **Description** | 为身份验证加密对称算法上下文设置（部署）密钥 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00042] Start

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00042 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24714 SWS\_CRYPT\_01808 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Start(ReadOnlyMemRegion iv=ReadOnlyMemRegion()) | |
| **Scope** | class ara::crypto::cryp::AuthCipherCtx | |
| **Syntax** | virtual ara::core::Result<void> Start (ReadOnlyMemRegion iv=ReadOnly MemRegion()) noexcept=0; | |
| **Parameters (in)** | iv | 可选的初始化向量 (IV) 或“nonce”值 |
| **Return value** | ara::core::Result<void> | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUninitialized Context | 如果上下文未初始化 |
| CryptoErrorDomain::kInvalidInputSize | 如果不支持提供的 IV 的大小（即如果它不足以进行初始化） |
| CryptoErrorDomain::kUnsupported | 如果基本算法（或其当前实现）主要不支持 IV 变量，但提供的 IV 值不为空，即 if (iv.empty() == false) |
| **Header file** | #include "ara/crypto/cryp/auth\_cipher\_ctx.h" | |
| **Description** | 初始化新数据处理或生成的上下文（取决于原语）。 如果 IV 大小大于算法支持的最大值，则实现可以仅使用序列中的前导字节。 | |
| **Additional** | - | |

(RS\_CRYPTO\_02302)

#### [SWRD-API-Crypto-00043] Start

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00043 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24715 SWS\_CRYPT\_01808 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Start(const SecretSeed &iv) | |
| **Scope** | class ara::crypto::cryp::AuthCipherCtx | |
| **Syntax** | virtual ara::core::Result<void> Start (const SecretSeed &iv) noexcept=0 | |
| **Parameters (in)** | iv | 初始化向量 (IV) 或“nonce”对象 |
| **Return value** | ara::core::Result<void> | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUninitialized Context | 如果上下文未初始化 |
| CryptoErrorDomain::kInvalidInputSize | 如果不支持提供的 IV 的大小（即如果它不足以进行初始化） |
| CryptoErrorDomain::kUnsupported | 如果基本算法（或其当前实现）主要不支持 IV 变量 |
| CryptoErrorDomain::kUsageViolation | 如果提供的 Secret Seed 对象的“允许使用”限制禁止此转换类型 |
| **Header file** | #include "ara/crypto/cryp/auth\_cipher\_ctx.h" | |
| **Description** | 初始化新数据处理或生成的上下文（取决于原语）。 如果 IV 大小大于算法支持的最大值，则实现可以仅使用序列中的前导字节。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00044] UpdateAssociatedData

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00044 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20312 SWS\_CRYPT\_01801 SWS\_CRYPT\_01802 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | UpdateAssociatedData(const RestrictedUseObject &in) | |
| **Scope** | class ara::crypto::cryp::AuthCipherCtx | |
| **Syntax** | virtual ara::core::Result<void> UpdateAssociatedData (const Restricted UseObject &in) noexcept=0; | |
| **Parameters (in)** | in | 应处理的输入消息的一部分 |
| **Return value** | ara::core::Result<void> | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Started | 如果摘要计算不是通过调用 Start() 方法启动的 |
| CryptoErrorDomain::kInvalidUsage Order | 如果 ProcessConfidentialData 已经被调用 |
| **Header file** | #include "ara/crypto/cryp/auth\_cipher\_ctx.h" | |
| **Description** | 通过指定的 RestrictedUseObject 更新摘要计算。 此方法专用于 RestrictedUseObject 是“消息”的一部分的情况。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00045] UpdateAssociatedData

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00045 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20313 SWS\_CRYPT\_01801 SWS\_CRYPT\_01802 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | UpdateAssociatedData(ReadOnlyMemRegion in) | |
| **Scope** | class ara::crypto::cryp::AuthCipherCtx | |
| **Syntax** | virtual ara::core::Result<void> UpdateAssociatedData (ReadOnlyMem Region in) noexcept=0; | |
| **Parameters (in)** | in | 应该处理的输入消息的一部分 |
| **Return value** | ara::core::Result<void> | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Started | 如果摘要计算不是通过调用 Start() 方法启动的 |
| CryptoErrorDomain::kInvalidUsage Order | 如果 ProcessConfidentialData 已经被调用 |
| **Header file** | #include "ara/crypto/cryp/auth\_cipher\_ctx.h" | |
| **Description** | 通过新的关联数据块更新摘要计算 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00046] UpdateAssociatedData

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00046 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20314 SWS\_CRYPT\_01801 SWS\_CRYPT\_01802 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | UpdateAssociatedData(std::uint8\_t in) | |
| **Scope** | class ara::crypto::cryp::AuthCipherCtx | |
| **Syntax** | virtual ara::core::Result<void> UpdateAssociatedData (std::uint8\_t in) noexcept=0; | |
| **Parameters (in)** | in | 作为输入消息一部分的字节值 |
| **Return value** | ara::core::Result<void> | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Started | 如果摘要计算不是通过调用 Start() 方法启动的 |
| CryptoErrorDomain::kInvalidUsage Order | 如果 ProcessConfidentialData 已经被调用 |
| **Header file** | #include "ara/crypto/cryp/auth\_cipher\_ctx.h" | |
| **Description** | 按指定字节更新摘要计算。 这种方法便于常量标签的处理。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00047] GetActualIvBitLength

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00047 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29035 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetActualIvBitLength(ara::core::Optional< CryptoObjectUid > ivUid) | |
| **Scope** | class ara::crypto::cryp::BlockService | |
| **Syntax** | virtual std::size\_t GetActualIvBitLength (ara::core::Optional< Crypto ObjectUid > ivUid) const noexcept=0; | |
| **Parameters (in)** | ivUid | 指向缓冲区的可选指针，用于保存现在加载到上下文中的 IV 对象的 COUID。 如果上下文由 SecretSeed 对象初始化，则输出缓冲区 \*ivUid 必须由加载的 IV 对象的 COUID 填充，在其他情况下，\*ivUid 必须由全零填充。 |
| **Return value** | std::size\_t | IV 的实际长度（现在设置为算法上下文）以位为单位 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/block\_service.h" | |
| **Description** | 获取加载到上下文的 IV 的实际位长 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00048] GetBlockSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00048 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29033 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetBlockSize() | |
| **Scope** | class ara::crypto::cryp::BlockService | |
| **Syntax** | virtual std::size\_t GetBlockSize () const noexcept=0; | |
| **Return value** | std::size\_t | 块的大小（以字节为单位） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/block\_service.h" | |
| **Description** | 获取基本算法的块（或内部缓冲区）大小。 | |
| **Additional** | - | |

(RS\_CRYPTO\_02309)

#### [SWRD-API-Crypto-00049] GetIvSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00049 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29032 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetIvSize() | |
| **Scope** | class ara::crypto::cryp::BlockService | |
| **Syntax** | virtual std::size\_t GetIvSize () const noexcept=0; | |
| **Return value:** | std::size\_t | IV 的默认预期大小（以字节为单位） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/block\_service.h" | |
| **Description** | 获取初始化向量 (IV) 或随机数的默认预期大小 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00050] IsValidIvSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00050 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29034 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | IsValidIvSize(std::size\_t ivSize) | |
| **Scope** | class ara::crypto::cryp::BlockService | |
| **Syntax** | virtual bool IsValidIvSize (std::size\_t ivSize) const noexcept=0; | |
| **Parameters (in)** | ivSize | IV 的长度（以字节为单位 |
| **Return value:** | bool | 如果算法支持提供的 IV 长度，则为 true，否则为 false |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/block\_service.h" | |
| **Description** | 验证特定初始化向量 (IV) 长度的有效性。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00051] ~CryptoContext

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00051 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20401 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | function |
| **Symbol** | ~CryptoContext() |
| **Scope** | class ara::crypto::cryp::CryptoContext |
| **Syntax** | virtual ~CryptoContext () noexcept=default; |
| **Exception Safety** | noexcept |
| **Header file** | #include "ara/crypto/cryp/crypto\_context.h" |
| **Description** | 析构函数 |
| **Additional** | - |

#### [SWRD-API-Crypto-00052] GetCryptoPrimitiveId

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00052 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20411 SWS\_CRYPT\_03904 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetCryptoPrimitiveId() | |
| **Scope** | class ara::crypto::cryp::CryptoContext | |
| **Syntax** | virtual CryptoPrimitiveId::Uptr GetCryptoPrimitiveId () const noexcept=0; | |
| **Return value** | CryptoPrimitiveId::Uptr | - |
| **Exception Safety** | noexcept | |
| **Header file** | #include "ara/crypto/cryp/crypto\_context.h" | |
| **Description** | 返回包含实例标识的 CryptoPrimtivId 实例。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00053] IsInitialized

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00053 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20412 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | IsInitialized() | |
| **Scope** | class ara::crypto::cryp::CryptoContext | |
| **Syntax** | virtual bool IsInitialized () const noexcept=0; | |
| **Return value** | bool | 如果加密上下文已完全初始化并准备好使用，则为 true，否则为 false |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/crypto\_context.h" | |
| **Description** | 检查加密上下文是否已经初始化并准备好使用。 它检查所有必需的值，包括：密钥值、IV/种子等。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00054] operator=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00054 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30214 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator=(const CryptoContext &other) | |
| **Scope** | class ara::crypto::cryp::CryptoContext | |
| **Syntax** | CryptoContext& operator= (const CryptoContext &other)=default; | |
| **Parameters (in)** | other | the other instance |
| **Return value** | CryptoContext & | \*this, containing the contents of other |
| **Header file** | #include "ara/crypto/cryp/crypto\_context.h" | |
| **Description** | 将另一个 CryptoContext 复制分配给此实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00055] operator=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00055 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30215 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator=(CryptoContext &&other) | |
| **Scope** | class ara::crypto::cryp::CryptoContext | |
| **Syntax** | CryptoContext& operator= (CryptoContext &&other)=default | |
| **Parameters (in)** | other | the other instance |
| **Return value** | CryptoContext & | \*this, containing the contents of other |
| **Header file** | #include "ara/crypto/cryp/crypto\_context.h" | |
| **Description** | 将另一个 CryptoContext 移动分配给该实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00056] MyProvider

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00056 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20654 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | MyProvider() | |
| **Scope** | class ara::crypto::cryp::CryptoContext | |
| **Syntax** | virtual CryptoProvider& MyProvider () const noexcept=0; | |
| **Return value** | CryptoProvider & | 对提供此上下文的 Crypto Provider 实例的引用 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/crypto\_context.h" | |
| **Description** | 获取对此上下文的 Crypto Provider 的引用。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00057] ~CryptoObject()

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00057 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20503 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | function |
| **Symbol** | ~CryptoObject() |
| **Scope** | class ara::crypto::cryp::CryptoObject |
| **Syntax** | virtual ~CryptoObject () noexcept=default; |
| **Exception Safety** | noexcept |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_object.h" |
| **Description** | 析构函数 |
| **Additional** | - |

#### [SWRD-API-Crypto-00058] Downcast

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00058 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20518 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Downcast(CryptoObject::Uptrc &&object) | |
| **Scope** | class ara::crypto::cryp::CryptoObject | |
| **Syntax** | template <class ConcreteObject> static ara::core::Result<typename ConcreteObject::Uptrc> Downcast ( CryptoObject::Uptrc &&object) noexcept; | |
| **Template param** | ConcreteObject | 向下转换的目标类型（从 CryptoObject 派生） |
| **Parameters (in)** | object | 指向常量通用加密对象接口的唯一智能指针 |
| **Return value** | ara::core::Result< typename Concrete Object::Uptrc > | 指向指定派生类型的向下转换常量接口的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kBadObjectType | 如果对象的实际类型不是指定的 ConcreteObject |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_object.h" | |
| **Description** | 向下转换并将唯一智能指针从通用 CryptoObject 接口移动到具体派生对象 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00059] GetCryptoPrimitiveId

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00059 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20505 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetCryptoPrimitiveId() | |
| **Scope** | class ara::crypto::cryp::CryptoObject | |
| **Syntax** | virtual CryptoPrimitiveId::Uptr GetCryptoPrimitiveId () const noexcept=0; | |
| **Return value** | CryptoPrimitiveId::Uptr | - |
| **Exception Safety** | noexcept | |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_object.h" | |
| **Description** | 返回此 CryptoObject 的 CryptoPrimtivId | |
| **Additional** | - | |

(RS\_CRYPTO\_02005)

#### [SWRD-API-Crypto-00060] GetObjectId

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00060 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20514 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetObjectId() | |
| **Scope** | class ara::crypto::cryp::CryptoObject | |
| **Syntax** | virtual COIdentifier GetObjectId () const noexcept=0; | |
| **Return value** | COIdentifier | 对象的 COIdentifier，包括对象的类型和 COUID（如果此对象不可识别，则为空 COUID） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_object.h" | |
| **Description** | 返回对象的 COIdentifier，其中包括对象的类型和 UID。 没有分配 COUID 的对象不能（安全地）序列化/导出或保存到非易失性存储。 如果一个对象同时是临时的且不可导出的，则该对象不应具有 COUID 一些不同类型的相关对象可以共享一个 COUID（例如私钥和公钥），但 COUID 和对象类型的组合必须始终是唯一的！ | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00061] GetPayloadSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00061 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20516 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetPayloadSize() | |
| **Scope** | class ara::crypto::cryp::CryptoObject | |
| **Syntax** | virtual std::size\_t GetPayloadSize () const noexcept=0; | |
| **Return value** | std::size\_t | 存储所需的对象有效负载的大小（以字节为单位） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_object.h" | |
| **Description** | 返回对象有效负载的实际大小。 返回值必须始终小于或等于此原始和对象类型预期的最大有效负载大小，可通过调用获得：MyProvider().GetPayloadStorageSize(GetObjectType(), GetPrimitiveId()).Value(); 返回值不考虑对象的元信息属性，但它们的大小是固定的，并且对于所有加密对象都是通用的，与它们的实际类型无关。 在分配 TrustedContainer 期间，加密提供者（和密钥存储提供者）根据其实现细节自动为对象的元信息保留空间。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00062] HasDependence

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00062 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20515 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | HasDependence() | |
| **Scope** | class ara::crypto::cryp::CryptoObject | |
| **Syntax** | virtual COIdentifier HasDependence () const noexcept=0; | |
| **Return value** | COIdentifie | 现有依赖的目标 COIdentifier 或 CryptoObjectType::kUnknown 和空 COUID，如果当前对象不依赖于另一个 CryptoObject |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_object.h" | |
| **Description** | 返回此 CryptoObject 所依赖的 CryptoObject 的 COIdentifier。 对于签名对象，此方法必须返回对对应签名验证公钥的引用！ CryptoObject 的明确标识需要两个组件：CryptoObjectUid 和 CryptoObjectType | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00063] IsExportable

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00063 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20513 SWS\_CRYPT\_40958 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | IsExportable() | |
| **Scope** | class ara::crypto::cryp::CryptoObjec | |
| **Syntax** | virtual bool IsExportable () const noexcept=0; | |
| **Return value** | bool | 如果对象是可导出的（即，如果它可以在 Crypto Provider 的受信任环境之外导出），则为 true |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_object.h" | |
| **Description** | 获取加密对象的可导出性属性。 可导出对象必须具有分配的 COUID（请参阅 GetObjectId()）。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00064] IsSession

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00064 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20512 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | IsSession() | |
| **Scope** | class ara::crypto::cryp::CryptoObject | |
| **Syntax** | virtual bool IsSession () const noexcept=0; | |
| **Return value** | bool | 如果对象是临时对象，则为 true（即其生命周期仅受当前会话限制） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_object.h" | |
| **Description** | 返回对象的“会话”（或“临时”）属性。 临时对象不能保存到 IOInterface 指向的持久存储位置！ 一个临时对象将与此接口实例一起被安全销毁！ 非会话对象必须具有分配的 COUID（请参阅 GetObjectId()）。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00065] Save

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00065 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20517 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Save(IOInterface &container) | |
| **Scope** | class ara::crypto::cryp::CryptoObject | |
| **Syntax** | virtual ara::core::Result<void> Save (IOInterface &container) const noexcept=0; | |
| **Parameters (in)** | container | 表示底层存储的 IOInterface |
| **Return value** | ara::core::Result<void> | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncompatible Object | 如果对象是“临时的”，但 IOInterface 代表一个 KeySlot。 |
| CryptoErrorDomain::kContent Restrictions | 如果对象不满足槽限制 |
| CryptoErrorDomain::kInsufficient Capacity | 如果目标容器的容量不够，即 if (container.Capacity() < this->StorageSize()) |
| CryptoErrorDomain::kModified Resource | 如果IOInterface打开后底层资源被修改，即IOInterface已经失效 |
| CryptoErrorDomain::kUnreserved Resource | 如果接口未打开可写。 |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_object.h" | |
| **Description** | 将自身保存到提供的 IOInterface ，具有属性“会话”的 CryptoObject 不能保存在 KeySlot 中 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00066] operator=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00066 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30208 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator=(const CryptoObject &other) | |
| **Scope** | class ara::crypto::cryp::CryptoObjec | |
| **Syntax** | CryptoObject& operator= (const CryptoObject &other)=default; | |
| **Parameters (in)** | other | the other instance |
| **Return value** | CryptoObject & | \*this, containing the contents of other |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_object.h" | |
| **Description** | 将另一个 CryptoObject 复制分配给该实例。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00067] operator=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00067 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30209 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator=(CryptoObject &&other) | |
| **Scope** | class ara::crypto::cryp::CryptoObject | |
| **Syntax** | CryptoObject& operator= (CryptoObject &&other)=default; | |
| **Parameters (in)** | other | the other instance |
| **Return value:** | CryptoObject & | \*this, containing the contents of other |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_object.h" | |
| **Description** | 将另一个 CryptoObject 移动分配给该实例。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00068] ~CryptoPrimitiveId

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00068 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10808 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | function |
| **Symbol** | ~CryptoPrimitiveId() |
| **Scope** | class ara::crypto::cryp::CryptoPrimitiveId |
| **Syntax** | virtual ~CryptoPrimitiveId () noexcept=default; |
| **Exception Safety** | noexcept |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_primitive\_id.h" |
| **Description** | 析构函数 |
| **Additional** | - |

#### [SWRD-API-Crypto-00069] GetPrimitiveId

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00069 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20652 SWS\_CRYPT\_03906 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetPrimitiveId() | |
| **Scope** | class ara::crypto::cryp::CryptoPrimitiveId | |
| **Syntax** | virtual AlgId GetPrimitiveId () const noexcept=0； | |
| **Return value** | AlgId | 二进制 Crypto Primitive ID |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_primitive\_id.h" | |
| **Description** | 获取原语的供应商特定 ID。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00070] GetPrimitiveName

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00070 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20651 SWS\_CRYPT\_03905 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetPrimitiveName() | |
| **Scope** | class ara::crypto::cryp::CryptoPrimitiveId | |
| **Syntax** | virtual const ara::core::StringView GetPrimitiveName () const noexcept=0; | |
| **Return value** | const ara::core::StringView | 加密原语的统一名称 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_primitive\_id.h" | |
| **Description** | 获取原语的统一名称。 可以完全或部分指定加密原语名称（有关详细信息，请参阅“加密原语命名约定”）。 返回的 StringView 实例的生命周期不应超过此 CryptoPrimitiveId 实例的生命周期！ | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00071] operator=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00071 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30212 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator=(const CryptoPrimitiveId &other) | |
| **Scope** | class ara::crypto::cryp::CryptoPrimitiveId | |
| **Syntax** | CryptoPrimitiveId& operator= (const CryptoPrimitiveId &other)=default | |
| **Parameters (in)** | other | the other instance |
| **Return value** | CryptoPrimitiveId& | \*this, containing the contents of other |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_primitive\_id.h" | |
| **Description** | 将另一个 CryptoPrimitiveId 复制分配给此实例。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00072] operator=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00072 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30213 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator=(CryptoPrimitiveId &&other) | |
| **Scope** | class ara::crypto::cryp::CryptoPrimitiveId | |
| **Syntax** | CryptoPrimitiveId& operator= (CryptoPrimitiveId &&other)=default; | |
| **Parameters (in)** | other | the other instance |
| **Return value** | CryptoPrimitiveId& | \*this, containing the contents of other |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_primitive\_id.h" | |
| **Description** | 将另一个 CryptoPrimitiveId 移动分配给此实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00073] AllocVolatileContainer

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00073 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20726 SWS\_CRYPT\_04208 SWS\_CRYPT\_40959 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | AllocVolatileContainer(std::size\_t capacity=0) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<VolatileTrustedContainer::Uptr> Alloc VolatileContainer (std::size\_t capacity=0) noexcept=0; | |
| **Parameters (in)** | capacity | 此易失性可信容器所需的容量（以字节为单位） |
| **Return value** | ara::core::Result<Volatile Trusted Container::Uptr> | 指向已分配易失性可信容器的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 根据直接指定的容量分配一个 Volatile（虚拟）Trusted Container。 Volatile Trusted Container 可用于执行导入操作。 当前进程获得分配的 Container 的“所有者”权限。 如果 (容量 == 0) 则容器的容量将根据支持的加密对象的最大大小自动选择。 一些易失性（临时）容器可以同时共存，而不会相互影响。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00074] AllocVolatileContainer

|  |  |  |  |
| --- | --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00074 | | |
| **Type** | Valid | | |
| **Priority** | H | | |
| **Upstream ID** | SWS\_CRYPT\_20727 SWS\_CRYPT\_04208 SWS\_CRYPT\_40959 | | |
| **CR** | - | | |
| **Consistency** | Yes | | |
| **Change Type** | 新增 | | |
| ***Kind*** | function | | |
| **Symbol** | AllocVolatileContainer(std::pair< AlgId, CryptoObjectType > theObjectDef) | | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | | |
| **Syntax** | virtual ara::core::Result<VolatileTrustedContainer::Uptr> Alloc VolatileContainer (std::pair< AlgId, CryptoObjectType > theObjectDef) noexcept=0; | | |
| **Parameters (in)** | theObjectDef | | 可以存储到此易失性可信容器的对象列表 |
| **Return value** | ara::core::Result< VolatileTrusted Container::Uptr > | | 指向已分配的易失性可信容器的唯一智能指针 |
| **Exception Safety** | noexcept | | |
| **Thread Safety** | Thread-safe | | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 如果列表中存在不受支持的对象类型和算法 ID 组合 | |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | | |
| **Description** | 根据间接指定托管任何列出的对象所需的最小容量，分配易失性（虚拟）可信容器。易失性可信容器可用于执行导入操作。当前进程获取已分配容器的"所有者"权限。实际容器容量计算为所有所列对象的最大存储大小。 | | |
| **Additional** | - | | |

#### [SWRD-API-Crypto-00075] ConvertToAlgId

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00075 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20711 SWS\_CRYPT\_40970 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ConvertToAlgId(ara::core::StringView primitiveName) | |
| **Scope** | class ara::crypto::cryp::CryptoProvid | |
| **Syntax** | virtual AlgId ConvertToAlgId (ara::core::StringView primitiveName) const noexcept=0; | |
| **Parameters (in)** | primitiveName | 加密原语的统一名称（有关更多详细信息，请参阅"加密基元命名约定"） |
| **Return value** | AlgId | 供应商特定的二进制算法 ID 或 kAlgIUndefined（如果不支持具有提供的名称的基元） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 将加密算法的公用名转换为相应的供应商特定的二进制算法 ID。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00076] ConvertToAlgName

|  |  |  |  |
| --- | --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00076 | | |
| **Type** | Valid | | |
| **Priority** | H | | |
| **Upstream ID** | SWS\_CRYPT\_20712 SWS\_CRYPT\_40971 | | |
| **CR** | - | | |
| **Consistency** | Yes | | |
| **Change Type** | 新增 | | |
| ***Kind*** | function | | |
| **Symbol** | ConvertToAlgName(AlgId algId) | | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | | |
| **Syntax** | virtual ara::core::Result<ara::core::String> ConvertToAlgName (AlgId algId) const noexcept=0; | | |
| **Parameters (in)** | algId | | 供应商特定的二进制算法 ID |
| **Return value** | ara::core::Result< ara::core::String > | | 加密算法的通用名称（有关详细信息，请参阅“加密原语命名约定”） |
| **Exception Safety** | noexcept | | |
| **Thread Safety** | Thread-safe | | |
| **Errors** | CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值 | |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | | |
| **Description** | 将供应商特定的二进制算法 ID 转换为加密算法的对应通用名称 | | |
| **Additional** | - | | |

#### [SWRD-API-Crypto-00077] CreateAuthCipherCtx

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00077 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20745 SWS\_CRYPT\_01806 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateAuthCipherCtx(AlgId algId) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<AuthCipherCtx::Uptr> CreateAuthCipherCtx ( AlgId algId) noexcept=0 | |
| **Parameters (in)** | algId | 目标加密算法的标识 |
| **Return value** | ara::core::Result< AuthCipherCtx::Uptr > | 指向已创建上下文的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 如果 algId 参数指定了不同于对称认证流密码的加密算法 |
| CryptoErrorDomain::kInvalidArgument | - |
| CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 创建对称认证密码上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00078] CreateDecryptorPrivateCtx

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00078 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20751 SWS\_CRYPT\_02700 SWS\_CRYPT\_02701 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateDecryptorPrivateCtx(AlgId algId) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<DecryptorPrivateCtx::Uptr> CreateDecryptor PrivateCtx (AlgId algId) noexcept=0; | |
| **Parameters (in)** | algId | 目标非对称加密/解密算法的标识符 |
| **Return value** | ara::core::Result< DecryptorPrivate Ctx::Uptr > | 指向已创建上下文的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 如果 algId 参数指定了不同于非对称加密/解密的加密算法 |
| CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 创建解密私钥上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00079] CreateEncryptorPublicCtx

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00079 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20750 SWS\_CRYPT\_40966 SWS\_CRYPT\_02700 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateEncryptorPublicCtx(AlgId algId) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<EncryptorPublicCtx::Uptr> CreateEncryptor PublicCtx (AlgId algId) noexcept=0; | |
| **Parameters (in)** | algId | 目标非对称加密/解密算法的标识符 |
| **Return value** | ara::core::Result<EncryptorPublicCtx::Uptr> | 指向已创建上下文的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 如果 algId 参数指定了不同于非对称加密/解密的加密算法 |
| CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 创建加密公钥上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00080] CreateHashDigest

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00080 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20761 SWS\_CRYPT\_40960 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateHashDigest(AlgId hashAlgId, ReadOnlyMemRegion value) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<Signature::Uptrc> CreateHashDigest (AlgId hashAlgId, ReadOnlyMemRegion value) noexcept=0; | |
| **Parameters (in)** | hashAlgId | 应用哈希函数密码算法的标识符 |
| value | 哈希摘要的原始 BLOB 值 |
| **Return value** | ara::core::Result<Signature::Uptrc> | 指向创建的 Signature 对象的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUnknown Identifier | 如果 hashAlgId 参数具有不受支持的值 |
| CryptoErrorDomain::kInvalidArgument | 如果 hashAlgId 参数指定不同于散列函数的加密算法 |
| CryptoErrorDomain::kInvalidInputSize | 如果 value 参数的大小无效（即与 hashAlgId 参数不兼容） |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 从直接提供的散列摘要组件构造签名对象。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00081] CreateHashFunctionCtx

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00081 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20747 SWS\_CRYPT\_00901 SWS\_CRYPT\_00902 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateHashFunctionCtx(AlgId algId) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<HashFunctionCtx::Uptr> CreateHashFunctionCtx (AlgId algId) noexcept=0; | |
| **Parameters (in)** | algId | 目标加密算法的标识符 |
| **Return value** | ara::core::Result< HashFunction Ctx::Uptr > | 指向已创建上下文的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 如果 algId 参数指定了不同于散列函数的加密算法 |
| CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 创建哈希函数上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00082] CreateKeyAgreementPrivateCtx

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00082 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20758 SWS\_CRYPT\_40969 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateKeyAgreementPrivateCtx(AlgId algId) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<KeyAgreementPrivateCtx::Uptr> CreateKey AgreementPrivateCtx (AlgId algId) noexcept=0; | |
| **Parameters (in)** | algId | 目标密钥协商加密算法的标识符 |
| **Return value** | ara::core::Result< KeyAgreement PrivateCtx::Uptr | 指向已创建上下文的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 如果 algId 参数指定了不同于密钥协议的加密算法 |
| CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 创建密钥协商私钥上下文。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00083] CreateKeyDecapsulatorPrivateCtx

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00083 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20753 SWS\_CRYPT\_40968 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateKeyDecapsulatorPrivateCtx(AlgId algId) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<KeyDecapsulatorPrivateCtx::Uptr> CreateKey DecapsulatorPrivateCtx (AlgId algId) noexcept=0; | |
| **Parameters (in)** | algId | 目标 KEM 加密算法的标识符 |
| **Return value** | ara::core::Result< KeyDecapsulator PrivateCtx::Uptr > | 指向已创建上下文的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 如果 algId 参数指定了不同于非对称 KEM 的加密算法 |
| CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 创建密钥封装机制 (KEM) 的密钥解封装器私钥上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00084] CreateKeyDerivationFunctionCtx

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00084 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20748 SWS\_CRYPT\_00601 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateKeyDerivationFunctionCtx(AlgId algId) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<KeyDerivationFunctionCtx::Uptr> CreateKey DerivationFunctionCtx (AlgId algId) noexcept=0; | |
| **Parameters (in)** | algId | 目标加密算法的标识符 |
| **Return value** | ara::core::Result< KeyDerivation FunctionCtx::Uptr > | 指向已创建上下文的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 如果 algId 参数指定了不同于密钥派生函数的加密算法 |
| CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 创建密钥派生函数上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00085] CreateKeyEncapsulatorPublicCtx

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00085 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20752 SWS\_CRYPT\_40967 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateKeyEncapsulatorPublicCtx(AlgId algId) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<KeyEncapsulatorPublicCtx::Uptr> CreateKey EncapsulatorPublicCtx (AlgId algId) noexcept=0; | |
| **Parameters (in)** | algId | 目标 KEM 加密算法的标识符 |
| **Return value** | ara::core::Result< KeyEncapsulator PublicCtx::Uptr > | 指向已创建上下文的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 如果 algId 参数指定了不同于非对称 KEM 的加密算法 |
| CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 创建密钥封装机制 (KEM) 的密钥封装器公钥上下文。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00086] CreateMessageAuthCodeCtx

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00086 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20746 SWS\_CRYPT\_01200 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateMessageAuthCodeCtx(AlgId algId) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<MessageAuthnCodeCtx::Uptr> CreateMessageAuth CodeCtx (AlgId algId) noexcept=0 | |
| **Parameters (in)** | algId | 目标加密算法的标识符 |
| **Return value** | ara::core::Result< MessageAuthnCode Ctx::Uptr | 指向已创建上下文的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 如果 algId 参数指定了不同于对称消息认证码的加密算法 |
| CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 创建对称消息认证代码上下文。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00087] CreateMsgRecoveryPublicCtx

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00087 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20755 SWS\_CRYPT\_02410 SWS\_CRYPT\_02411 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateMsgRecoveryPublicCtx(AlgId algId) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<MsgRecoveryPublicCtx::Uptr> CreateMsg RecoveryPublicCtx (AlgId algId) noexcept=0; | |
| **Parameters (in)** | algId | 目标非对称加密算法的标识符 |
| **Return value** | ara::core::Result< MsgRecoveryPublic Ctx::Uptr > | 指向已创建上下文的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 如果 algId 参数指定了不同于带有消息恢复的非对称签名编码的加密算法 |
| CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 创建消息恢复公钥上下文。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00088] CreateRandomGeneratorCtx

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00088 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20741 SWS\_CRYPT\_00500 SWS\_CRYPT\_00501 SWS\_CRYPT\_00506 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateRandomGeneratorCtx(AlgId algId=kAlgIdDefault, bool initialize=true) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<RandomGeneratorCtx::Uptr> CreateRandom GeneratorCtx (AlgId algId=kAlgIdDefault, bool initialize=true) noexcept=0; | |
| **Parameters (in)** | algId | 目标RNG算法的标识符。 如果没有给出 algId，则返回默认的 RNG |
| initialize | 指示返回的上下文是否应由堆栈初始化（即播种） |
| **Return value** | ara::core::Result< RandomGenerator Ctx::Uptr > | 指向创建的 RNG 上下文的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kBusyResource | if (initialize == true) 但当前无法播种上下文（例如，由于缺乏熵） |
| CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值，或者如果 (alg Id == kAlgIdDefault) 并且 CryptoProvider 不提供任何 RandomGeneratorCtx |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 创建一个随机数生成器 (RNG) 上下文。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00089] CreateSigEncodePrivateCtx

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00089 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20754 SWS\_CRYPT\_02409 SWS\_CRYPT\_02413 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateSigEncodePrivateCtx(AlgId algId) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<SigEncodePrivateCtx::Uptr> CreateSigEncode PrivateCtx (AlgId algId) noexcept=0; | |
| **Parameters (in)** | algId | 目标非对称加密算法的标识符 |
| **Return value** | ara::core::Result< SigEncodePrivate Ctx::Uptr > | 指向已创建上下文的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 如果 algId 参数指定了不同于带有消息恢复的非对称签名编码的加密算法 |
| CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 创建签名编码私钥上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00090] CreateSignature

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00090 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20760 SWS\_CRYPT\_40961 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateSignature(AlgId signAlgId, ReadOnlyMemRegion value, const RestrictedUseObject &key, AlgId hashAlgId=kAlgIdNone) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<Signature::Uptrc> CreateSignature (AlgId signAlgId, ReadOnlyMemRegion value, const RestrictedUseObject &key, AlgId hashAlgId=kAlgIdNone) noexcept=0; | |
| **Parameters (in)** | signAlgId | 应用签名/MAC/AE/AEAD 加密算法的标识符 |
| value | 签名/MAC 的原始 BLOB 值 |
| key | 用于签名或 MAC/AE/AEAD 操作的对称或非对称密钥（根据 signAlg Id） |
| hashAlgId | 与签名算法一起应用的哈希函数算法的标识符 |
| **Return value** | ara::core::Result< Signature::Uptrc > | 指向创建的 Signature 对象的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncompatible Arguments | 如果 signAlgId 和 hashAlgId 参数指定不兼容的算法（如果 signAlgId 包括哈希函数规范），或者如果与 key 参数关联的加密原语与提供的 signAlgId 或 hashAlgId 参数不兼容 |
| CryptoErrorDomain::kInvalidInputSize | 如果 value 参数的大小无效（即与 signAlgId 参数不兼容） |
| CryptoErrorDomain::kInvalidArgument | 如果 signAlgId 或 hashAlgId 参数分别指定不同于签名/MAC/AE/AEAD 和消息摘要的加密算法 |
| CryptoErrorDomain::kUnknown Identifier | 如果 signAlgId 或 hashAlgId 参数具有不受支持的值 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 从直接提供的数字签名/MAC 或经过身份验证的加密 (AE/AEAD) 的组件构造签名对象。 数字签名 BLOB 值中的所有整数始终以 Big Endian 字节顺序呈现（即 MSF - 最高有效字节优先） | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00091] CreateSignerPrivateCtx

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00091 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20756 SWS\_CRYPT\_02408 SWS\_CRYPT\_02413 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateSignerPrivateCtx(AlgId algId) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<SignerPrivateCtx::Uptr> CreateSignerPrivate Ctx (AlgId algId) noexcept=0; | |
| **Parameters (in)** | algId | 目标签名加密算法的标识符 |
| **Return value** | ara::core::Result< SignerPrivate Ctx::Uptr > | 指向已创建上下文的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 如果 algId 参数指定了不同于私钥签名的加密算法 |
| CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 创建签名私钥上下文。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00092] CreateStreamCipherCtx

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00092 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20744 SWS\_CRYPT\_40964 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateStreamCipherCtx (AlgId algId) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<StreamCipherCtx::Uptr> CreateStreamCipherCtx (AlgId algId) noexcept=0; | |
| **Parameters (in)** | algId | 目标加密算法的标识符 |
| **Return value** | ara::core::Result< StreamCipher Ctx::Uptr > | 指向已创建上下文的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 如果 algId 参数指定了不同于对称流密码的加密算法 |
| CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 创建对称流加密上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00093] CreateSymmetricBlockCipherCtx

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00093 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20742 SWS\_CRYPT\_40963 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateSymmetricBlockCipherCtx(AlgId algId) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<SymmetricBlockCipherCtx::Uptr> Create SymmetricBlockCipherCtx (AlgId algId) noexcept=0; | |
| **Parameters (in)** | algId | 目标加密算法的标识符 |
| **Return value** | ara::core::Result< SymmetricBlock CipherCtx::Uptr > | 指向已创建上下文的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 创建对称块加密上下文 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-00094] CreateSymmetricKeyWrapperCtx

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00094 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20743 SWS\_CRYPT\_40965 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateSymmetricKeyWrapperCtx(AlgId algId) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<SymmetricKeyWrapperCtx::Uptr> Create SymmetricKeyWrapperCtx (AlgId algId) noexcept=0; | |
| **Parameters (in)** | algId | 目标加密算法的标识符 |
| **Return value** | ara::core::Result< SymmetricKey WrapperCtx::Uptr > | 指向已创建上下文的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 如果 algId 参数指定了不同于对称密钥包装的加密算法 |
| CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 创建对称密钥包装算法上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00095] CreateVerifierPublicCtx

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00095 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20757 SWS\_CRYPT\_02400 SWS\_CRYPT\_02414 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CreateVerifierPublicCtx(AlgId algId) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<VerifierPublicCtx::Uptr> CreateVerifier PublicCtx (AlgId algId) noexcept=0; | |
| **Parameters (in)** | algId | 目标签名加密算法的标识符 |
| **Return value** | ara::core::Result< VerifierPublic Ctx::Uptr> | 指向已创建上下文的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 如果 algId 参数指定了不同于公钥签名验证的加密算法 |
| CryptoErrorDomain::kUnknown Identifier | 如果 algId 参数具有不受支持的值 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 创建签名验证公钥上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00096] ~CryptoProvider

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00096 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20710 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | function |
| **Symbol** | ~CryptoProvider() |
| **Scope** | class ara::crypto::cryp::CryptoProvider |
| **Syntax** | virtual ~CryptoProvider () noexcept=default; |
| **Exception Safety** | noexcept |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" |
| **Description** | 析构函数 |
| **Additional** |  |

#### [SWRD-API-Crypto-00097] ExportPublicObject

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00097 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20731 SWS\_CRYPT\_04203 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ExportPublicObject(const IOInterface &container, Serializable::FormatId formatId=Serializable::kFormatDefault) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > Export PublicObject (const IOInterface &container, Serializable::FormatId formatId=Serializable::kFormatDefault) noexcept=0; | |
| **Parameters (in)** | container | 包含要导出的对象的 IOInterface |
| formatId | 输出格式的 CryptoProvider 特定标识符 |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 序列化数据所需的实际容量 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kEmptyContainer | 如果容器是空的 |
| CryptoErrorDomain::kUnexpected Value | 如果容器包含秘密加密对象 |
| CryptoErrorDomain::kInsufficient Capacity | if (serialized.empty() == false)，但它的容量不足以存储结果 |
| CryptoErrorDomain::kModified Resource | 如果IOInterface打开后底层资源被修改，即IOInterface已经失效 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 从 IOInterface 公开导出对象（即无需中间创建加密对象）。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00098] ExportSecuredObject

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00098 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20728 SWS\_CRYPT\_04202 SWS\_CRYPT\_04213 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ExportSecuredObject(const CryptoObject &object, SymmetricKeyWrapperCtx &transport Context) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > Export SecuredObject (const CryptoObject &object, SymmetricKeyWrapperCtx &transportContext) noexcept=0; | |
| **Parameters (in)** | object | 用于导出的加密对象 |
| transportContex | 由传输密钥初始化的对称密钥包装上下文（允许的用法：kAllowKeyExporting） |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 包装的加密对象数据 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncompatible Object | 如果由于 IsExportable() 返回 false 而无法导出对象 |
| CryptoErrorDomain::kIncompleteArg State | 如果 transportContext 未初始化 |
| CryptoErrorDomain::kIncompatible Object | 如果加载到 transportContext 的密钥没有必需的属性（注意：这是此方法的可选错误条件） |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 以安全的方式导出加密对象。 if (serialized.empty() == true) 则该方法仅返回所需的大小，但 transportContext 的内容保持新增！ 只能导出可导出且已完成的对象（即具有 UUID） | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00099] ExportSecuredObject

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00099 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20729 SWS\_CRYPT\_04202 SWS\_CRYPT\_04213 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ExportSecuredObject(const IOInterface &container, SymmetricKeyWrapperCtx &transport Context) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > Export SecuredObject (const IOInterface &container, SymmetricKeyWrapperCtx &transportContext) noexcept=0; | |
| **Parameters (in)** | container | 关联要导出的对象的 IOInterface |
| transportContext | 由传输密钥初始化的对称密钥包装上下文（允许的用法：kAllowKeyExporting） |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 序列化数据所需的实际容量 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kEmptyContainer | 如果容器是空的 |
| CryptoErrorDomain::kInsufficient Capacity | 如果序列化缓冲区的大小不足以保存输出数据 |
| CryptoErrorDomain::kIncompleteArg State | transportContext没有被初始化 |
| CryptoErrorDomain::kIncompatible Object | 如果加载到 transportContext 的密钥没有必需的属性（注意：这是此方法的可选错误条件） |
| CryptoErrorDomain::kModified Resource | 如果IOInterface打开后底层资源被修改，即IOInterface已经失效 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 直接从 IOInterface 安全地导出对象（即无需中间创建加密对象）。 if (serialized == nullptr) 则该方法仅返回所需的大小，但 transportContext 的内容保持新增。 此方法可用于重新导出刚刚导入的对象但在另一个传输密钥上。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00100] GeneratePrivateKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00100 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20722 SWS\_CRYPT\_40962 SWS\_CRYPT\_03300 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GeneratePrivateKey(AlgId algId, AllowedUsageFlags allowedUsage, bool isSession=false, bool isExportable=false) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<PrivateKey::Uptrc> GeneratePrivateKey (AlgId algId, AllowedUsageFlags allowedUsage, bool isSession=false, bool is Exportable=false) noexcept=0; | |
| **Parameters (in)** | algId | 目标公私钥密码算法的标识符 |
| allowedUsag | 定义可以使用目标密钥的允许转换类型列表的标志（请参阅 RestrictedUseObject 范围内的常量） |
| isSession | 目标键的“会话”（或“临时”）属性（如果为true） |
| isExportable | 目标键的可导出性属性（如果为true） |
| **Return value** | ara::core::Result< PrivateKey::Uptrc > | 指向创建的私钥对象的智能唯一指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUnknown Identifier | 如果 algId 具有不受支持的值 |
| CryptoErrorDomain::kIncompatible Arguments | 如果 allowedUsage 参数与目标算法 algId 不兼容（注意：这是此方法的可选错误条件） |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 分配一个新的对应类型的私钥上下文并随机生成密钥值。 应该为私钥和公钥共享一个公共 COUID。 任何可序列化（即可保存/非会话或可导出）密钥都必须生成自己的 COUID！ | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00101] GenerateSeed

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00101 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20723 SWS\_CRYPT\_40962 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GenerateSeed(AlgId algId, SecretSeed::Usage allowedUsage, bool isSession=true, bool is Exportable=false) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<SecretSeed::Uptrc> GenerateSeed (AlgId alg Id, SecretSeed::Usage allowedUsage, bool isSession=true, bool is Exportable=false) noexcept=0; | |
| **Parameters (in)** | algId | 目标加密算法的标识符 |
| allowedUsage | 定义可以使用目标种子的允许转换类型列表的滞后（请参阅 RestrictedUseObject 范围内的常量） |
| isSession | 目标种子的“会话”（或“临时”）属性（如果为true） |
| isExportable | 目标种子的可出口性属性（如果为true） |
| **Return value** | ara::core::Result< SecretSeed::Uptrc > | 指向生成的 SecretSeed 对象的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUnknown Identifier | 如果 algId 具有不受支持的值 |
| CryptoErrorDomain::kIncompatible Arguments | 如果 allowedUsage 参数与目标算法 algId 不兼容（注意：这是此方法的可选错误条件） |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 生成请求算法的随机秘密种子对象。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00102] GenerateSymmetricKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00102 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20721 SWS\_CRYPT\_40962 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GenerateSymmetricKey(AlgId algId, AllowedUsageFlags allowedUsage, bool isSession=true, bool isExportable=false) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<SymmetricKey::Uptrc> GenerateSymmetricKey ( AlgId algId, AllowedUsageFlags allowedUsage, bool isSession=true, bool isExportable=false) noexcept=0; | |
| **Parameters (in)** | algId | 目标对称密码算法的标识符 |
| allowedUsage | 定义可以使用目标密钥的允许转换类型列表的标志（请参阅 RestrictedUseObject 范围内的常量） |
| isSession | 目标密钥的“会话”（或“临时”）属性（如果为true） |
| isExportable | 目标密钥的可导出性属性（如果为true） |
| **Return value** | ara::core::Result< Symmetric Key::Uptrc > | 指向创建的对称密钥对象的智能唯一指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUnknown Identifier | 如果 algId 具有不受支持的值 |
| CryptoErrorDomain::kIncompatible Arguments | 如果 allowedUsage 参数与目标算法 algId 不兼容（注意：这是此方法的可选错误条件 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 分配一个新的对称密钥对象并用一个新的随机生成的值填充它。 任何可序列化（即可保存/非会话或可导出）密钥都必须生成自己的 COUID！ 默认情况下，Crypto Provider 应使用所有受支持的 RNG（理想情况下为 TRNG）的最佳内部实例。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00103] GetSerializedSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00103 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20724 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetSerializedSize(CryptoObjectType cryptoObjectType, AlgId algId, Serializable::FormatId formatId=Serializable::kFormatDefault) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<std::size\_t> GetSerializedSize (CryptoObject Type cryptoObjectType, AlgId algId, Serializable::FormatId format Id=Serializable::kFormatDefault) const noexcept=0; | |
| **Parameters (in)** | cryptoObjectType | 目标对象的类型 |
| algId | 目标对象的 Crypto Provider 算法 ID |
| formatId | 输出格式的 Crypto Provider 特定标识符 |
| **Return value** | ara::core::Result< std::size\_t > | 存储以指定格式序列化的对象所需的大小 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUnknown Identifier | 如果任何参数具有不受支持的值 |
| CryptoErrorDomain::kIncompatible Arguments | 如果任何一对参数不兼容 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 返回特定格式的对象序列化所需的缓冲区大小 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00104] ImportPublicObject

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00104 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20732 SWS\_CRYPT\_04205 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ImportPublicObject(IOInterface &container, ReadOnlyMemRegion serialized, CryptoObject Type expectedObject=CryptoObjectType::kUndefined) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<void> ImportPublicObject (IOInterface &container, ReadOnlyMemRegion serialized, CryptoObjectType expected Object=CryptoObjectType::kUndefined) noexcept=0; | |
| **Parameters (in)** | serialized | 包含应导入到 IOInterface 的安全序列化对象的内存区域 |
| expectedObject | 预期的对象类型（默认值 CryptoObjectType::kUnknown 表示不检查 |
| **Parameters (out)** | container | 用于存储导入对象的 IOInterface |
| **Return value** | ara::core::Result<void> | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUnexpected Value | 如果序列化包含不正确的数据 |
| CryptoErrorDomain::kBadObjectType | if (expectedObject != CryptoObjectType::kUnknown)，但实际对象类型与预期不同 |
| CryptoErrorDomain::kInsufficient Capacity | 如果容器的容量不足以保存反序列化的对象 |
| CryptoErrorDomain::kModified Resource | 如果IOInterface打开后底层资源被修改，即IOInterface已经失效。 |
| CryptoErrorDomain::kUnreserved Resource | 如果 IOInterface 未打开可写。 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 将公共序列化对象导入 IOInterface 指向的存储位置以进行后续处理（不分配加密对象）。 如果 (expectedObject != CryptoObjectType::k Unknown) 并且实际对象类型与预期不同，则此方法失败。 如果序列化包含不正确的数据，则此方法失败。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00105] ImportSecuredObject

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00105 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20730 SWS\_CRYPT\_04204 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ImportSecuredObject(IOInterface &container, ReadOnlyMemRegion serialized, SymmetricKey WrapperCtx &transportContext, bool isExportable=false, CryptoObjectType expected Object=CryptoObjectType::kUndefined) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<void> ImportSecuredObject (IOInterface &container, ReadOnlyMemRegion serialized, SymmetricKeyWrapperCtx &transportContext, bool isExportable=false, CryptoObjectType expected Object=CryptoObjectType::kUndefined) noexcept=0; | |
| **Parameters (in)** | serialized | 包含应导入到 IOInterface 的安全序列化对象的内存区域 |
| transportContext | 由传输密钥初始化的对称密钥包装上下文（允许的用法：kAllowKeyImporting） |
| isExportable | 目标对象的可导出性属性 |
| expectedObject | 预期的对象类型（默认值 CryptoObjectType::kUnknown 表示不检查 |
| **Parameters (out)** | container | 用于存储导入对象的 IOInterface |
| **Return value** | ara::core::Result<void> | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUnexpected Value | 如果序列化包含不正确的数据 |
| CryptoErrorDomain::kBadObjectType | if (expectedObject != CryptoObjectType::kUnknown)，但实际对象类型与预期不同 |
| CryptoErrorDomain::kIncompleteArg State | 如果 transportContext 未初始化 |
| CryptoErrorDomain::kIncompatible Object | 如果加载到 transportContext 的密钥没有必需的属性（注意：这是此方法的可选错误条件） |
| CryptoErrorDomain::kInsufficient Capacity | 如果容器的容量不足以保存反序列化的对象 |
| CryptoErrorDomain::kModified Resource | 如果IOInterface打开后底层资源被修改，即IOInterface已经失效。 |
| CryptoErrorDomain::kUnreserved Resource | 如果接口未打开可写 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 将安全序列化的对象导入到由 IOInterface 表示的持久性或易失性存储中，以进行后续处理。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00106] LoadObject

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00106 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20733 SWS\_CRYPT\_04200 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | LoadObject(const IOInterface &container) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<CryptoObject::Uptrc> LoadObject (const IOInterface &container) noexcept=0; | |
| **Parameters (in)** | container | 包含要加载的加密对象的 IOInterface |
| **Return value** | ara::core::Result< CryptoObject::Uptrc > | 指向已创建对象的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kEmptyContainer | 如果容器是空的 |
| CryptoErrorDomain::kResourceFault | 如果容器内容物损坏 |
| CryptoErrorDomain::kModified Resource | 如果IOInterface打开后底层资源被修改，即IOInterface已经失效 |
| CryptoErrorDomain::kIncompatible Object | 如果底层资源属于另一个不兼容的 CryptoProvider |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 从提供的 IOInterface 加载任何加密对象。 | |
| **Notes** | 此方法是 CryptoProvider 和 Key Storage Provider 之间的“绑定”方法之一。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00107] LoadPrivateKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00107 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20764 SWS\_CRYPT\_04200 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | LoadPrivateKey(const IOInterface &container) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<PrivateKey::Uptrc> LoadPrivateKey (const IOInterface &container) noexcept=0; | |
| **Parameters (in)** | container | 包含要加载的加密对象的 IOInterface |
| **Return value** | ara::core::Result< PrivateKey::Uptrc > | 指向 PrivateKey 的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Errors** | CryptoErrorDomain::kEmptyContainer | 如果容器是空的 |
| CryptoErrorDomain::kResourceFault | 如果容器内容被毁坏 |
| CryptoErrorDomain::kModified Resource | 如果IOInterface打开后底层资源被修改，即IOInterface已经失效 |
| CryptoErrorDomain::kIncompatible Object | 如果底层资源属于另一个不兼容的 CryptoProvider |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 从提供的 IOInterface 加载私钥。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00108] LoadPublicKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00108 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20763 SWS\_CRYPT\_04200 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | LoadPublicKey(const IOInterface &container) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<PublicKey::Uptrc> LoadPublicKey (const IOInterface &container) noexcept=0; | |
| **Parameters (in)** | container | 包含要加载的加密对象的 IOInterface |
| **Return value** | ara::core::Result< PublicKey::Uptrc > | 指向 PublicKey 的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Errors** | CryptoErrorDomain::kEmptyContainer | 如果容器是空的 |
| CryptoErrorDomain::kResourceFault | 如果容器内容被毁坏 |
| CryptoErrorDomain::kModified Resource | 如果IOInterface打开后底层资源被修改，即IOInterface已经失效 |
| CryptoErrorDomain::kIncompatible Object | 如果底层资源属于另一个不兼容的 CryptoProvider |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 从提供的 IOInterface 加载公钥。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00109] LoadSecretSeed

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00109 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20765 SWS\_CRYPT\_04200 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | LoadSecretSeed(const IOInterface &container) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<SecretSeed::Uptrc> LoadSecretSeed (const IOInterface &container) noexcept=0; | |  |
| **Parameters (in)** | container | 包含要加载的加密对象的 IOInterface |
| **Return value** | ara::core::Result< SecretSeed::Uptrc> | 指向 SecretSeed的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Errors** | CryptoErrorDomain::kEmptyContainer | 如果容器是空的 |
| CryptoErrorDomain::kResourceFault | 如果容器内容被毁坏 |
| CryptoErrorDomain::kModified Resource | 如果IOInterface打开后底层资源被修改，即IOInterface已经失效 |
| CryptoErrorDomain::kIncompatible Object | 如果底层资源属于另一个不兼容的 CryptoProvider |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 从提供的 IOInterface 加载秘密种子。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00110] LoadSymmetricKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00110 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20762 SWS\_CRYPT\_04200 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | LoadSymmetricKey(const IOInterface &container) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<SymmetricKey::Uptrc> LoadSymmetricKey (const IOInterface &container) noexcept=0; | |  |
| **Parameters (in)** | container | 包含要加载的加密对象的 IOInterface |
| **Return value** | ara::core::Result< Symmetric Key::Uptrc > | 指向 SymmetricKey的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Errors** | CryptoErrorDomain::kEmptyContainer | 如果容器是空的 |
| CryptoErrorDomain::kResourceFault | 如果容器内容被毁坏 |
| CryptoErrorDomain::kModified Resource | 如果IOInterface打开后底层资源被修改，即IOInterface已经失效 |
| CryptoErrorDomain::kIncompatible Object | 如果底层资源属于另一个不兼容的 CryptoProvider |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 从提供的 IOInterface 加载对称密钥。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00111] GetBlockSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00111 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29023 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetBlockSize() | |
| **Scope** | class ara::crypto::cryp::CryptoService | |
| **Syntax** | virtual std::size\_t GetBlockSize () const noexcept=0; | |
| **Return value** | std::size\_t | 块的大小（以字节为单位） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/crypto\_service.h" | |
| **Description** | 获取基本算法的块（或内部缓冲区）大小。 对于摘要、字节流密码和 RNG 上下文，它是一种提供信息的方法，仅用于优化接口使用。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00112] GetMaxInputSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00112 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29021 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetMaxInputSize(bool suppressPadding=false) | |
| **Scope** | class ara::crypto::cryp::CryptoService | |
| **Syntax** | virtual std::size\_t GetMaxInputSize (bool suppressPadding=false) const noexcept=0; | |
| **Parameters (in)** | suppressPadding | 如果为true，则该方法计算纯数据块的整个空间仅用于有效负载的情况的大小 |
| **Return value** | std::size\_t | 输入数据块的最大大小（以字节为单位） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/crypto\_service.h" | |
| **Description** | 获取输入数据块的最大预期大小。 suppressPadding 参数，它将等于块大小。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00113] GetMaxOutputSize 接口

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00113 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29022 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetMaxOutputSize(bool suppressPadding=false) | |
| **Scope** | class ara::crypto::cryp::CryptoService | |
| **Syntax** | virtual std::size\_t GetMaxOutputSize (bool suppressPadding=false) const noexcept=0; | |
| **Parameters (in)** | suppressPadding | 如果为true，则该方法计算纯数据块的整个空间仅用于有效负载的情况的大小 |
| **Return value** | std::size\_t | 输出数据块的最大大小（以字节为单位） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/crypto\_service.h" | |
| **Description** | 获取输出数据块的最大可能大小。 如果 (IsEncryption() == true) 则此方法返回的值独立于 suppressPadding 参数，并且将等于块大小 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00114] operator=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00114 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30216 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator=(const CryptoProvider &other) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | CryptoProvider& operator= (const CryptoProvider &other)=default; | |
| **Parameters (in)** | other | the other instance |
| **Return value** | CryptoProvider & | \*this, containing the contents of other |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 将另一个 CryptoProvider 复制分配给此实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00115] operator=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-ID (SWRD-ID 编号规则参见附录A-信息定义需求ID。) | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30217 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator=(CryptoProvider &&other) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | CryptoProvider& operator= (CryptoProvider &&other)=default | |
| **Parameters (in)** | other | the other instance |
| **Return value** | CryptoProvider & | \*this, containing the contents of other |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 将另一个 CryptoProvider 移动分配给该实例。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00116] GetCryptoService

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00116 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20802 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetCryptoService() | |
| **Scope** | class ara::crypto::cryp::DecryptorPrivateCtx | |
| **Syntax** | virtual CryptoService::Uptr GetCryptoService () const noexcept=0; | |
| **Return value** | CryptoService::Uptr | - |
| **Exception Safety** | noexcept | |
| **Header file** | #include "ara/crypto/cryp/decryptor\_private\_ctx.h" | |
| **Description** | 获取 CryptoService 实例。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00117] ProcessBlock

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00117 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20812 SWS\_CRYPT\_02705 SWS\_CRYPT\_02706  SWS\_CRYPT\_02726 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ProcessBlock(ReadOnlyMemRegion in, bool suppressPadding=false) | |
| **Scope** | class ara::crypto::cryp::DecryptorPrivateCtx | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > Process Block (ReadOnlyMemRegion in, bool suppressPadding=false) const noexcept=0; | |
| **Parameters (in)** | in | 输入数据块 |
| suppressPadding | 如果为真，则该方法不应用填充，但有效负载应填充整个纯数据块 |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 输出数据的实际大小（它总是 <= out.size()），如果输入数据块的内容不正确，则为 0 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncorrectInput Size | 如果违反了上述关于输入大小的规则 |
| CryptoErrorDomain::kInsufficient Capacity | 如果 out.size() 不足以存储转换结果 |
| CryptoErrorDomain::kUninitialized Context | 如果上下文未由密钥值初始化 |
| **Header file** | #include "ara/crypto/cryp/decryptor\_private\_ctx.h" | |
| **Description** | 根据加密配置处理（加密/解密）输入块。 加密时 (suppressPadding == true) 期望：in.size() == GetMaxInputSize(true) && out.size() >= GetMaxOutputSize(true)。 加密时 (suppressPadding == false)期望：in.size() <= GetMaxInputSize(false) && in.size() > 0 && out.size() >= GetMaxOutputSize(false)。 解密期望：in.size() == GetMaxInputSize() && out.size() >= GetMaxOutput Size(suppressPadding)。 只有当你严格确定输出数据的大小时，才应谨慎使用 (out.size() < GetMaxOutputSize()) ！ 在 (suppress Padding == true) 的情况下，纯文本的实际大小应该等于纯数据块的完整大小（由算法定义）！ | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00118] ProcessBlock

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00118 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20813 SWS\_CRYPT\_02705 SWS\_CRYPT\_02706  SWS\_CRYPT\_02726 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ProcessBlock(ReadOnlyMemRegion in, bool suppressPadding=false) | |
| **Scope** | class ara::crypto::cryp::DecryptorPrivateCtx | |
| **Syntax** | template <typename Alloc = <implementation-defined>> ara::core::Result<ByteVector<Alloc> > ProcessBlock (ReadOnlyMemRegion in, bool suppressPadding=false) const noexcept; | |
| **Template param** | Alloc | 输出容器的自定义分配器类型 |
| **Parameters (in)** | in | 输入数据块 |
| suppressPadding | 如果为true，则该方法不应用填充，但有效负载应填充整个纯数据块 |
| **Return value** | ara::core::Result< ByteVector< Alloc > > | 输出块的托管容器 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncorrectInput Size | 如果违反了上述关于输入大小的规则 |
| CryptoErrorDomain::kInsufficient Capacity | 如果 out.size() 不足以存储转换结果 |
| CryptoErrorDomain::kUninitialized Context | 如果上下文未由密钥值初始化 |
| **Header file** | #include "ara/crypto/cryp/decryptor\_private\_ctx.h" | |
| **Description** | 根据加密配置处理（加密/解密）输入块。 此方法根据实际保存的值设置输出容器的大小！ 加密时 (suppressPadding == true) 的期望：in.size() == GetMaxInputSize(true) && out.capacity() >= GetMaxOutputSize(true)。 加密时 (suppressPadding == false) 的期望：in.size() <= GetMaxInputSize(false) && in.size() > 0 && out.capacity() >= GetMaxOutput Size(false)。 解密期望：in.size() == GetMaxInputSize() && out.capacity() >= Get MaxOutputSize(suppressPadding)。 仅当你严格确定输出数据的大小时，才应谨慎使用 (out.capacity() < GetMaxOutputSize())！ 在 (suppressPadding == true) 的情况下，纯文本的实际大小应该等于纯数据块的完整大小（由算法定义）！ | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00119] Reset

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00119 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20811 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Reset() | |
| **Scope** | class ara::crypto::cryp::DecryptorPrivateCtx | |
| **Syntax** | virtual ara::core::Result<void> Reset () noexcept=0; | |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/decryptor\_private\_ctx.h" | |
| **Description** | 清除加密上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00120] SetKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00120 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20810 SWS\_CRYPT\_02702 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SetKey(const PrivateKey &key) | |
| **Scope** | class ara::crypto::cryp::DecryptorPrivateCtx | |
| **Syntax** | virtual ara::core::Result<void> SetKey (const PrivateKey &key) noexcept=0; | |
| **Parameters (in)** | key | 源密钥对象 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncompatible Object | 如果提供的密钥对象与此对称密钥上下文不兼容 |
| CryptoErrorDomain::kUsageViolation | 如果与此上下文关联的转换类型被提供的密钥对象的“允许使用”限制禁止 |
| **Header file** | #include "ara/crypto/cryp/decryptor\_private\_ctx.h" | |
| **Description** | 设置（部署）解密器私有算法上下文的密钥 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00121] Compare

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00121 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29013 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Compare(ReadOnlyMemRegion expected, std::size\_t offset=0) | |
| **Scope** | class ara::crypto::cryp::DigestService | |
| **Syntax** | virtual ara::core::Result<bool> Compare (ReadOnlyMemRegion expected, std::size\_t offset=0) const noexcept=0; | |
| **Parameters (in)** | expected | 包含预期摘要值的内存区域 |
|  | offset | 计算摘要中第一个字节的位置，用于比较开始 |
| **Return value** | ara::core::Result< bool > | 如果预期的字节序列与计算摘要相同，则为 true |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Finished | 如果未通过调用 Finish() 方法完成摘要计算 |
| CryptoErrorDomain::kBruteForceRisk | 如果缓冲的摘要属于 MAC/HMAC/AE/AEAD 上下文，该上下文由未经 kAllowSignature 许可的密钥初始化，但请求的摘要的实际大小小于 8 字节（这是对暴力攻击的保护） |
| **Header file** | #include "ara/crypto/cryp/digest\_service.h" | |
| **Description** | 将计算的摘要与预期值进行比较。 整个摘要值保存在上下文中直到下一次调用 Start()，因此它的任何部分都可以再次验证或提取。 if (full\_digest\_size <= offset) || (expected.size() == 0) 然后返回 false； 否则 compare\_size = min(expected.size(), (full\_digest\_size - offset)) 字节。 此方法可以在函数 ara::core::memcmp() 标准化后实现为“内联”。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00122] GetDigestSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00122 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29012 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetDigestSize() | |
| **Scope** | class ara::crypto::cryp::DigestService | |
| **Syntax** | virtual std::size\_t GetDigestSize () const noexcept=0; | |
| **Return value** | std::size\_t | 此摘要函数的完整输出的大小（以字节为单位） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/digest\_service.h" | |
| **Description** | 获取输出摘要大小 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00123] IsFinished

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00123 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29015 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | IsFinished() | |
| **Scope** | class ara::crypto::cryp::DigestService | |
| **Syntax** | virtual bool IsFinished () const noexcept=0; | |
| **Return value** | bool | 如果先前启动的流处理通过调用 Finish() 或 FinishBytes() 方法完成，则为 true |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/digest\_service.h" | |
| **Description** | 检查流处理的当前状态：完成或否。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00124] IsStarted

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00124 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29014 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | IsStarted() | |
| **Scope** | class ara::crypto::cryp::DigestService | |
| **Syntax** | virtual bool IsStarted () const noexcept=0; | |
| **Return value** | bool | 如果处理是通过调用 Start() 方法开始并且尚未完成，则为 true |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/digest\_service.h" | |
| **Description** | 检查流处理的当前状态：已启动或未启动。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00125] GetCryptoService

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00125 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21002 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetCryptoService() | |
| **Scope** | class ara::crypto::cryp::EncryptorPublicCtx | |
| **Syntax** | virtual CryptoService::Uptr GetCryptoService () const noexcept=0; | |
| **Return value** | CryptoService::Uptr | - |
| **Exception Safety** | noexcept | |
| **Header file** | #include "ara/crypto/cryp/encryptor\_public\_ctx.h" | |
| **Description** | 获取 CryptoService 实例。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00126] ProcessBlock

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00126 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20812 SWS\_CRYPT\_02704 SWS\_CRYPT\_02706  SWS\_CRYPT\_02726 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ProcessBlock(ReadOnlyMemRegion in, bool suppressPadding=false) | |
| **Scope** | class ara::crypto::cryp::EncryptorPrivateCtx | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > ProcessBlock (ReadOnlyMemRegion in, bool suppressPadding=false) const noexcept=0; | |
| **Parameters (in)** | in | 输入数据块 |
| suppressPadding | 如果为真，则该方法不应用填充，但有效负载应填充整个纯数据块 |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 输出数据的实际大小（它总是 <= out.size()），如果输入数据块的内容不正确，则为 0 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncorrectInput Size | 如果违反了上述关于输入大小的规则 |
| CryptoErrorDomain::kUninitialized Context | 如果上下文未由密钥值初始化 |
| **Header file** | #include "ara/crypto/cryp/encryptor\_public\_ctx.h" | |
| **Description** | 根据加密配置处理（加密/解密）输入块。 加密时 (suppressPadding == true) 期望：in.size() == GetMaxInputSize(true) && out.size() >= GetMaxOutputSize(true)。 加密时 (suppressPadding == false)期望：in.size() <= GetMaxInputSize(false) && in.size() > 0 && out.size() >= GetMaxOutputSize(false)。 解密期望：in.size() == GetMaxInputSize() && out.size() >= GetMaxOutput Size(suppressPadding)。 只有当你严格确定输出数据的大小时，才应谨慎使用 (out.size() < GetMaxOutputSize()) ！ 在 (suppress Padding == true) 的情况下，纯文本的实际大小应该等于纯数据块的完整大小（由算法定义）！ | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00127] ProcessBlock

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00127 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21013 SWS\_CRYPT\_02704 SWS\_CRYPT\_02706  SWS\_CRYPT\_02726 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ProcessBlock(ReadOnlyMemRegion in, bool suppressPadding=false) | |
| **Scope** | class ara::crypto::cryp::EncryptorPublicCtx | |
| **Syntax** | template <typename Alloc = <implementation-defined>> ara::core::Result<ByteVector<Alloc> > ProcessBlock (ReadOnlyMemRegion in, bool suppressPadding=false) const noexcept; | |
| **Template param** | Alloc | 输出容器的自定义分配器类型 |
| **Parameters (in)** | in | 输入数据块 |
| suppressPadding | 如果为true，则该方法不应用填充，但有效负载应填充整个纯数据块 |
| **Return value** | ara::core::Result< ByteVector< Alloc > > | 输出块的托管容器 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncorrectInput Size | 如果违反了上述关于输入大小的规则 |
| CryptoErrorDomain::kInsufficient Capacity | 如果 out.size() 不足以存储转换结果 |
| CryptoErrorDomain::kUninitialized Context | 如果上下文未由密钥值初始化 |
| **Header file** | #include "ara/crypto/cryp/encryptor\_public\_ctx.h" | |
| **Description** | 根据加密配置处理（加密/解密）输入块。 此方法根据实际保存的值设置输出容器的大小！ 加密时 (suppressPadding == true) 的期望：in.size() == GetMaxInputSize(true) && out.capacity() >= GetMaxOutputSize(true)。 加密时 (suppressPadding == false) 的期望：in.size() <= GetMaxInputSize(false) && in.size() > 0 && out.capacity() >= GetMaxOutput Size(false)。 解密期望：in.size() == GetMaxInputSize() && out.capacity() >= Get MaxOutputSize(suppressPadding)。 仅当你严格确定输出数据的大小时，才应谨慎使用 (out.capacity() < GetMaxOutputSize())！ 在 (suppressPadding == true) 的情况下，纯文本的实际大小应该等于纯数据块的完整大小（由算法定义）！ | |
| **Additional** |  | |

#### [SWRD-API-Crypto-00128] Reset

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00128 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21011 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Reset() | |
| **Scope** | class ara::crypto::cryp::EncryptorPublicCtx | |
| **Syntax** | virtual ara::core::Result<void> Reset () noexcept=0; | |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/encryptor\_public\_ctx.h" | |
| **Description** | 清除加密上下文 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-00129] SetKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00129 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21010 SWS\_CRYPT\_02703 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SetKey(const PrivateKey &key) | |
| **Scope** | class ara::crypto::cryp::EncryptorPublicCtx | |
| **Syntax** | virtual ara::core::Result<void> SetKey (const PublicKey &key) noexcept=0; | |
| **Parameters (in)** | key | 源密钥对象 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncompatible Object | 如果提供的密钥对象与此对称密钥上下文不兼容 |
| CryptoErrorDomain::kUsageViolation | 如果与此上下文关联的转换类型被提供的密钥对象的“允许使用”限制禁止 |
| **Header file** | #include "ara/crypto/cryp/encryptor\_public\_ctx.h" | |
| **Description** | 设置（部署）解密器私有算法上下文的密钥 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00130 ]~ExtensionService

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00130 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_29041 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | function |
| **Symbol** | ~ExtensionService() |
| **Scope** | class ara::crypto::cryp::ExtensionService |
| **Syntax** | virtual ~ExtensionService () noexcept=default; |
| **Exception Safety** | noexcept |
| **Header file** | #include "ara/crypto/cryp/extension\_service.h" |
| **Description** | 析构函数 |
| **Additional** | - |

#### [SWRD-API-Crypto-00131] GetActualKeyBitLength

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00131 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29045 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetActualKeyBitLength() | |
| **Scope** | class ara::crypto::cryp::ExtensionService | |
| **Syntax** | virtual std::size\_t GetActualKeyBitLength () const noexcept=0; | |
| **Return value** | std::size\_t | 密钥的实际长度（现在设置为算法上下文）以位为单位 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/extension\_service.h" | |
| **Description** | 获取加载到上下文中的密钥的实际位长。 如果尚未为上下文设置密钥，则返回 0。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00132] GetActualKeyCOUID

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00132 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29047 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetActualKeyCOUID() | |
| **Scope** | class ara::crypto::cryp::ExtensionService | |
| **Syntax** | virtual CryptoObjectUid GetActualKeyCOUID () const noexcept=0; | |
| **Return value** | CryptoObjectUid | CryptoObject 的 COUID |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/extension\_service.h" | |
| **Description** | 获取部署到此扩展服务所附加的上下文的密钥的 COUID。 如果还没有为上下文设置键，则返回一个空的 COUID (Nil)。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00133] GetAllowedUsage

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00133 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29046 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetAllowedUsage() | |
| **Scope** | class ara::crypto::cryp::ExtensionService | |
| **Syntax** | virtual AllowedUsageFlags GetAllowedUsage () const noexcept=0; | |
| **Return value** | AllowedUsageFlags | 指定上下文允许使用的位标志的组合 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/extension\_service.h" | |
| **Description** | 获取此上下文的允许用法（根据加载到此上下文的密钥对象属性）。 如果上下文尚未由密钥对象初始化，则必须返回零（所有标志均已重置）。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00134] GetMaxKeyBitLength

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00134 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29044 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetMaxKeyBitLength() | |
| **Scope** | class ara::crypto::cryp::ExtensionService | |
| **Syntax** | virtual std::size\_t GetMaxKeyBitLength () const noexcept=0; | |
| **Return value** | std::size\_t | 支持的最大密钥长度（以位为单位） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/extension\_service.h" | |
| **Description** | 获取最大支持的密钥长度（以位为单位）。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00135] GetMinKeyBitLength

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00135 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29043 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetMinKeyBitLength() | |
| **Scope** | class ara::crypto::cryp::ExtensionService | |
| **Syntax** | virtual std::size\_t GetMinKeyBitLength () const noexcept=0; | |
| **Return value** | std::size\_t | 密钥的最小支持长度（以位为单位） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/extension\_service.h" | |
| **Description** | 获取支持的最小密钥长度（以位为单位）。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00136] IsKeyBitLengthSupported

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00136 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29048 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | IsKeyBitLengthSupported(std::size\_t keyBitLength) | |
| **Scope** | class ara::crypto::cryp::ExtensionService | |
| **Syntax** | virtual bool IsKeyBitLengthSupported (std::size\_t keyBitLength) const noexcept=0; | |
| **Parameters (in)** | keyBitLength | 密钥的长度（以位为单位 |
| **Return value** | bool | 如果上下文支持提供的密钥长度值，则为 true |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/extension\_service.h" | |
| **Description** | 通过上下文验证对特定密钥长度的支持 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00137] IsKeyAvailable

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00137 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29049 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | IsKeyAvailable() | |
| **Scope** | class ara::crypto::cryp::ExtensionService | |
| **Syntax** | virtual bool IsKeyAvailable () const noexcept=0; | |
| **Return value** | bool | 如果没有设置密钥，则为 FALSE |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/extension\_service.h" | |
| **Description** | 检查是否已将密钥设置为此上下文。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00138] operator=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00138 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30218 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator=(const ExtensionService &other) | |
| **Scope** | class ara::crypto::cryp::ExtensionService | |
| **Syntax** | ExtensionService& operator= (const ExtensionService &other)=default; | |
| **Parameters (in)** | other | the other instance |
| **Return value** | ExtensionService & | \*this, containing the contents of other |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/extension\_service.h" | |
| **Description** | 将另一个 ExtensionService 复制分配给此实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00139] operator=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00139 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30219 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator=(ExtensionService &&other) | |
| **Scope** | class ara::crypto::cryp::ExtensionService | |
| **Syntax** | ExtensionService& operator= (ExtensionService &&other)=default; | |
| **Parameters (in)** | other | the other instance |
| **Return value** | ExtensionService & | \*this, containing the contents of other |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/extension\_service.h" | |
| **Description** | 将另一个 ExtensionService 移动分配给此实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00140] Finish

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00140 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21115 SWS\_CRYPT\_00906 SWS\_CRYPT\_00910 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Finish() | |
| **Scope** | class ara::crypto::cryp::HashFunctionCtx | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > Finish () noexcept=0; | |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 输出数据缓存 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Started | 如果摘要计算不是通过调用 Start() 方法启动的 |
| CryptoErrorDomain::kInvalidUsage Order | 如果摘要计算尚未开始或至少未更新一次 |
| **Header file** | #include "ara/crypto/cryp/hash\_function\_ctx.h" | |
| **Description** | 完成摘要计算并可选择生成“签名”对象。 只有在调用此方法后，才能对摘要进行签名、验证、提取或比较。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00141] GetDigestService

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00141 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21102 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetDigestService() | |
| **Scope** | class ara::crypto::cryp::HashFunctionCtx | |
| **Syntax** | virtual DigestService::Uptr GetDigestService () const noexcept=0; | |
| **Return value** | DigestService::Uptr | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/hash\_function\_ctx.h" | |
| **Description** | 获取 DigestService 实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00142] GetDigest

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00142 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21116 SWS\_CRYPT\_00907 SWS\_CRYPT\_00919 | |
| **CR** |  | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetDigest(std::size\_t offset=0) | |
| **Scope** | class ara::crypto::cryp::HashFunctionCtx | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > Get Digest (std::size\_t offset=0) const noexcept=0; | |
| **Parameters (in)** | offset | 应该放置到输出缓冲区的摘要的第一个字节的位置 |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 实际存储到输出缓冲区的摘要字节数（它们总是 <= output.size() 并在下面表示为 return\_size） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Finished | 如果未通过调用 Finish() 方法完成摘要计算 |
| **Header file** | #include "ara/crypto/cryp/hash\_function\_ctx.h" | |
| **Description** | 获取计算摘要的请求部分。 整个摘要值保存在上下文中直到下一次调用 Start()，因此它的任何部分都可以再次提取或验证。 如果 (full\_digest\_size <= offset) 那么 return\_size = 0 字节； 否则 return\_size = min(output.size(), (full\_digest\_size - offset)) 字节。 此方法可以在函数 ara::core::memcpy() 标准化后实现为“内联”。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00143] GetDigest

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00143 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21117 SWS\_CRYPT\_00907 SWS\_CRYPT\_00919 | |
| **CR** |  | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetDigest(std::size\_t offset=0) | |
| **Scope** | class ara::crypto::cryp::HashFunctionCtx | |
| **Syntax** | template <typename Alloc = <implementation-defined>> ara::core::Result<ByteVector<Alloc> > GetDigest (std::size\_t offset=0) const noexcept | |
| **Template param** | Alloc | 输出容器的自定义分配器类型 |
| **Parameters (in)** | offset | 应该放置到输出缓冲区的摘要的第一个字节的位置 |
| **Return value** | ara::core::Result< ByteVector< Alloc > > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Finished | 如果未通过调用 Finish() 方法完成摘要计算 |
| CryptoErrorDomain::kUsageViolation | 如果缓冲的摘要属于由没有 kAllowSignature许可的密钥初始化的 MAC/HMAC/AE/AEAD 上下文 |
| **Header file** | #include "ara/crypto/cryp/hash\_function\_ctx.h" | |
| **Description** | 将计算摘要的请求部分获取到预先保留的托管容器。 该方法根据实际保存的值设置输出容器的大小。 整个摘要值保存在上下文中直到下一次调用 Start()，因此它的任何部分都可以再次提取或验证。 如果 (full\_digest\_size <= offset) 那么 return\_size = 0 字节； 否则 return\_size = min(output.capacity(), (full\_digest\_size - offset)) 字节。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00144] Start

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00144 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21118 SWS\_CRYPT\_00908 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Start() | |
| **Scope** | class ara::crypto::cryp::HashFunctionCtx | |
| **Syntax** | virtual ara::core::Result<void> Start () noexcept=0; | |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kMissing Argument | 配置的散列函数需要一个 IV |
| **Header file** | #include "ara/crypto/cryp/hash\_function\_ctx.h" | |
| **Description** | 为没有 IV 的新数据流处理或生成（取决于原语）初始化上下文。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00145] Start

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00145 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21110 SWS\_CRYPT\_00908 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Start(ReadOnlyMemRegion iv) | |
| **Scope** | class ara::crypto::cryp::HashFunctionCtx | |
| **Syntax** | virtual ara::core::Result<void> Start (ReadOnlyMemRegion iv) noexcept=0; | |
| **Parameters (in)** | iv | 选的初始化向量 (IV) 或“nonce”值 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidInputSize | 如果不支持提供的 IV 的大小（即如果它不足以进行初始化） |
| CryptoErrorDomain::kUnsupported | 如果基本算法（或其当前实现）主要不支持 IV 变体，但提供的 IV 值不为空，即 if (iv.empty() == false) |
| **Header file** | #include "ara/crypto/cryp/hash\_function\_ctx.h" | |
| **Description** | 为新的数据流处理或生成初始化上下文（取决于原语）。 如果 IV 大小大于算法支持的最大值，则实现可以仅使用序列中的前导字节。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00146] Start

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00146 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21111 SWS\_CRYPT\_00908 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Start(const SecretSeed &iv) | |
| **Scope** | class ara::crypto::cryp::HashFunctionCtx | |
| **Syntax** | virtual ara::core::Result<void> Start (const SecretSeed &iv) noexcept=0; | |
| **Parameters (in)** | iv | 选的初始化向量 (IV) 或“nonce”值 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidInputSize | 如果不支持提供的 IV 的大小（即如果它不足以进行初始化） |
| CryptoErrorDomain::kUnsupported | 如果基本算法（或其当前实现）主要不支持 IV 变体，但提供的 IV 值不为空，即 if (iv.empty() == false) |
| **Header file** | #include "ara/crypto/cryp/hash\_function\_ctx.h" | |
| **Description** | 为新的数据流处理或生成初始化上下文（取决于原语）。 如果 IV 大小大于算法支持的最大值，则实现可以仅使用序列中的前导字节。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00147] Update

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00147 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21112 SWS\_CRYPT\_00905 SWS\_CRYPT\_00909 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Update(const RestrictedUseObject &in) | |
| **Scope** | class ara::crypto::cryp::HashFunctionCtx | |
| **Syntax** | virtual ara::core::Result<void> Update (const RestrictedUseObject &in) noexcept=0; | |
| **Parameters (in)** | in | 应处理的输入消息的一部分 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Started | 如果摘要计算不是通过调用 Start() 方法启动的 |
| **Header file** | #include "ara/crypto/cryp/hash\_function\_ctx.h" | |
| **Description** | 通过消息的新部分更新摘要计算上下文。 此方法专用于 RestrictedUseObject 是“消息”的一部分的情况。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00148] Update

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00148 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21113 SWS\_CRYPT\_00905 SWS\_CRYPT\_00909 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Update(ReadOnlyMemRegion in) | |
| **Scope** | class ara::crypto::cryp::HashFunctionCtx | |
| **Syntax** | virtual ara::core::Result<void> Update (ReadOnlyMemRegion in) noexcept=0; | |
| **Parameters (in)** | in | 应处理的输入消息的一部分 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Started | 如果摘要计算不是通过调用 Start() 方法启动的 |
| **Header file** | #include "ara/crypto/cryp/hash\_function\_ctx.h" | |
| **Description** | 通过消息的新部分更新摘要计算上下文。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00149] Update

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00149 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21114 SWS\_CRYPT\_00905 SWS\_CRYPT\_00909 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Update(std::uint8\_t in) | |
| **Scope** | class ara::crypto::cryp::HashFunctionCtx | |
| **Syntax** | virtual ara::core::Result<void> Update (std::uint8\_t in) noexcept=0; | |
| **Parameters (in)** | in | 作为输入消息一部分的字节值 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Started | 如果摘要计算不是通过调用 Start() 方法启动的 |
| **Header file** | #include "ara/crypto/cryp/hash\_function\_ctx.h" | |
| **Description** | 通过消息的新部分更新摘要计算上下文。 这种方法便于常量标签的处理。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00150] AgreeKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00150 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21312 SWS\_CRYPT\_03302 SWS\_CRYPT\_03303 SWS\_CRYPT\_03304 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | AgreeKey(const PublicKey &otherSideKey, CryptoAlgId targetAlgId, AllowedUsageFlags allowedUsage, ara::core::Optional< const KeyDerivationFunctionCtx::Uptr > kdf, ara::core::Optional< ReadOnlyMemRegion > salt, ara::core::Optional< ReadOnlyMemRegion > ctxLabel) | |
| **Scope** | class ara::crypto::cryp::KeyAgreementPrivateCtx | |
| **Syntax** | virtual ara::core::Result<SymmetricKey::Uptrc> AgreeKey (const Public Key &otherSideKey, CryptoAlgId targetAlgId, AllowedUsageFlags allowed Usage, ara::core::Optional< const KeyDerivationFunctionCtx::Uptr > kdf, ara::core::Optional< ReadOnlyMemRegion > salt, ara::core::Optional< ReadOnlyMemRegion > ctxLabel) const noexcept=0; | |
| **Parameters (in)** | otherSideKe | 密钥协商另一方的公钥 |
| targetAlgId | 目标对称算法的标识符（也定义了目标密钥长度） |
| allowedUsage | 目标密钥的允许使用范围 |
| kdf | 密钥派生函数的可选上下文，可用于目标密钥生成 |
| salt | 一个可选的盐值（如果使用，它应该对于目标密钥的每个实例都是唯一的） |
| ctxLabel | 一个可选的应用程序特定的“上下文标签”（它可以识别目标密钥和/或通信方的目的 |
| **Return value** | ara::core::Result< Symmetric Key::Uptrc > | 指向 SymmetricKey 对象的唯一指针，其中包含由密钥协商算法生成的计算共享秘密或密钥材料 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrc::kUninitializedContext | 如果上下文未由密钥值初始化 |
| CryptoErrc::kIncompatibleObject | 如果公钥和私钥对应不同的算法 |
| **Header file** | #include "ara/crypto/cryp/key\_agreement\_private\_ctx.h" | |
| **Description** | 通过在这个私钥和另一方的公钥之间执行密钥协商算法来产生一个公共的对称密钥。 生成的 SymmetricKey 对象具有以下属性：会话、不可导出。 此方法可用于直接生成目标密钥，而无需创建中间 SecretSeed 对象。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00151 AgreeSeed

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00151 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21311 SWS\_CRYPT\_03301 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | AgreeSeed(const PublicKey &otherSideKey, ara::core::Optional< AllowedUsageFlags > allowed Usage) | |
| **Scope** | class ara::crypto::cryp::KeyAgreementPrivateCtx | |
| **Syntax** | virtual ara::core::Result<SecretSeed::Uptrc> AgreeSeed (const Public Key &otherSideKey, ara::core::Optional< AllowedUsageFlags > allowed Usage) const noexcept=0; | |
| **Parameters (in)** | otherSideKey | 密钥协商另一方的公钥 |
| allowedUsage | 目标种子的允许使用范围 |
| **Return value** | ara::core::Result< SecretSeed::Uptrc > | 指向 SecretSeed 对象的唯一指针，其中包含由密钥协商算法生成的密钥材料 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrc::kUninitializedContext | 如果上下文未由密钥值初始化 |
| CryptoErrc::kIncompatibleObject | 如果公钥和私钥对应不同的算法 |
| **Header file** | #include "ara/crypto/cryp/key\_agreement\_private\_ctx.h" | |
| **Description** | 通过在这个私钥和另一方的公钥之间执行密钥协商算法产生一个公共秘密种子。 生成的 SecretSeed 对象具有以下属性：会话、不可导出、AlgID（此 Key-Agreement Algorithm ID）。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00152] GetExtensionService

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00152 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21302 SWS\_CRYPT\_03313 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetExtensionService() | |
| **Scope** | class ara::crypto::cryp::KeyAgreementPrivateCtx | |
| **Syntax** | virtual ExtensionService::Uptr GetExtensionService () const noexcept=0; | |
| **Return value** | ExtensionService::Uptr | - |
| **Exception Safety** | noexcept | |
| **Header file** | #include "ara/crypto/cryp/key\_agreement\_private\_ctx.h" | |
| **Description** | 获取 ExtensionService 实例。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00153] Reset

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00153 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21314 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Reset() | |
| **Scope** | class ara::crypto::cryp::KeyAgreementPrivateCtx | |
| **Syntax** | virtual ara::core::Result<void> Reset () noexcept=0; | |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/key\_agreement\_private\_ctx.h" | |
| **Description** | 清除加密上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00154] SetKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00154 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21313 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SetKey(const PrivateKey &key) | |
| **Scope** | class ara::crypto::cryp::KeyAgreementPrivateCtx | |
| **Syntax** | virtual ara::core::Result<void> SetKey (const PrivateKey &key) noexcept=0; | |
| **Parameters (in)** | key | 源密钥对象 |
| **Return value** | ara::core::Result< void > |  |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrc::kIncompatibleObject | 如果提供的密钥对象与此私钥上下文不兼容 |
| CryptoErrc::kUsageViolation | 如果与此上下文关联的转换类型被提供的密钥对象的“允许使用”限制禁止 |
| **Header file** | #include "ara/crypto/cryp/key\_agreement\_private\_ctx.h" | |
| **Description** | 设置（部署）密钥协商私有算法上下文的密钥 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00155] DecapsulateKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00155 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21412 SWS\_CRYPT\_03003 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | DecapsulateKey(ReadOnlyMemRegion input, CryptoAlgId keyingDataAlgId, KeyDerivation FunctionCtx &kdf, CryptoAlgId kekAlgId, ara::core::Optional< AllowedUsageFlags > allowed Usage) | |
| **Scope** | class ara::crypto::cryp::KeyDecapsulatorPrivateCtx | |
| **Syntax** | virtual ara::core::Result<SymmetricKey::Uptrc> DecapsulateKey (Read OnlyMemRegion input, CryptoAlgId keyingDataAlgId, KeyDerivation FunctionCtx &kdf, CryptoAlgId kekAlgId, ara::core::Optional< Allowed UsageFlags > allowedUsage) const noexcept=0; | |
| **Parameters (in)** | input | 输入缓冲区（其大小应等于 GetEncapsulatedSize() 字节） |
| keyingDataAlgId | 返回的对称密钥的算法 ID |
| kdf | 密钥派生函数的上下文，应用于 KEK 生产 |
| kekAlgId | KEK 的算法 ID |
| allowedUsage | 返回的对称密钥对象的允许使用范围（默认 = kAllowKdfMaterialAnyUsage） |
| **Return value** | ara::core::Result< Symmetric Key::Uptrc > | 从解封装的密钥数据中实例化的对称密钥对象的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrc::kUninitializedContext | 如果上下文不是由私钥值初始化的 |
| CryptoErrc::kInvalidArgument | 如果 kekAlgId 或 kdf 与此上下文不兼容 |
| CryptoErrc::kInvalidInputSiz | 如果此上下文不支持输入的大小 |
| **Header file** | #include "ara/crypto/cryp/key\_decapsulator\_private\_ctx.h" | |
| **Description** | 解封装密钥数据以用于后续处理（例如安全通信）。 生成的 SymmetricKey 对象具有以下属性：会话、不可导出。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00156] DecapsulateSeed

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00156 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21411 SWS\_CRYPT\_03004 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | DecapsulateSeed(ReadOnlyMemRegion input, ara::core::Optional< AllowedUsageFlags > allowedUsage) | |
| **Scope** | class ara::crypto::cryp::KeyDecapsulatorPrivateCtx | |
| **Syntax** | virtual ara::core::Result<SecretSeed::Uptrc> DecapsulateSeed (ReadOnly MemRegion input, ara::core::Optional< AllowedUsageFlags > allowed Usage) const noexcept=0; | |
| **Parameters (in)** | input | 带有封装种子的缓冲区（其大小应等于 GetEncapsulatedSize() 字节 |
| allowedUsage | 目标种子的允许使用范围（默认 = kAllowKdfMaterialAnyUsage） |
| **Return value** | ara::core::Result< SecretSeed::Uptrc > | 指向 SecretSeed 对象的唯一智能指针，它使密钥材料从输入缓冲区中解封装 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrc::kUninitializedContext | 如果上下文不是由私钥值初始化的 |
| CryptoErrc::kInvalidInputSize | if this context does not support the size of input |
| **Header file** | #include "ara/crypto/cryp/key\_decapsulator\_private\_ctx.h" | |
| **Description** | 解封装密钥材料。 生成的 SecretSeed 对象具有以下属性：会话、不可导出、AlgID = 此 KEM的AlgID。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00157] GetEncapsulatedSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00157 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21416 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetEncapsulatedSize() | |
| **Scope** | class ara::crypto::cryp::KeyDecapsulatorPrivateCtx | |
| **Syntax** | virtual std::size\_t GetEncapsulatedSize () const noexcept=0; | |
| **Return value** | std::size\_t | 封装数据块的大小（以字节为单位） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/key\_decapsulator\_private\_ctx.h" | |
| **Description** | 获取封装数据块的固定大小。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00158] GetExtensionService

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00158 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21402 SWS\_CRYPT\_03009 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetExtensionService() | |
| **Scope** | class ara::crypto::cryp::KeyDecapsulatorPrivateCtx | |
| **Syntax** | virtual ExtensionService::Uptr GetExtensionService () const noexcept=0; | |
| **Return value** | ExtensionService::Uptr | - |
| **Exception Safety** | noexcept | |
| **Header file** | #include "ara/crypto/cryp/key\_decapsulator\_private\_ctx.h" | |
| **Description** | 获取 ExtensionService 实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00159] GetKekEntropy

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00159 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21415 | |
| **CR** |  | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetKekEntropy() | |
| **Scope** | class ara::crypto::cryp::KeyDecapsulatorPrivateCtx | |
| **Syntax** | virtual std::size\_t GetKekEntropy () const noexcept=0; | |
| **Return value** | std::size\_t | KEK 材料的熵（以位为单位） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/key\_decapsulator\_private\_ctx.h" | |
| **Description** | 获取密钥加密密钥 (KEK) 材料的熵（位长）。 对于 RSA 系统，返回值对应于模数N 的长度（减 1）。 对于 DH 类系统，返回值对应于模数 q 的长度（减 1）。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00160] Reset

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00160 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21414 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Reset() | |
| **Scope** | class ara::crypto::cryp::KeyDecapsulatorPrivateCtx | |
| **Syntax** | virtual ara::core::Result<void> Reset () noexcept=0; | |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/key\_decapsulator\_private\_ctx.h" | |
| **Description** | 清除加密上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00161] SetKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00161 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21413 SWS\_CRYPT\_03005 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SetKey(const PrivateKey &key) | |
| **Scope** | class ara::crypto::cryp::KeyDecapsulatorPrivateCtx | |
| **Syntax** | virtual ara::core::Result<void> SetKey (const PrivateKey &key) noexcept=0; | |
| **Parameters (in)** | key | 源密钥对象 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrc::kIncompatibleObject | 如果提供的密钥对象与此私钥上下文不兼容 |
| CryptoErrc::kUsageViolation | 如果与此上下文关联的转换类型被提供的密钥对象的“允许使用”限制禁止 |
| **Header file** | #include "ara/crypto/cryp/key\_decapsulator\_private\_ctx.h" | |
| **Description** | 设置（部署）密钥解封装器私有算法上下文的密钥 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00162] AddSalt

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00162 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21512 SWS\_CRYPT\_00608 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | AddSalt(ReadOnlyMemRegion salt) | |
| **Scope** | class ara::crypto::cryp::KeyDerivationFunctionCtx | |
| **Syntax** | virtual ara::core::Result<void> AddSalt (ReadOnlyMemRegion salt) noexcept=0 | |
| **Parameters (in)** | salt | 一个盐值（如果使用，它应该对于目标密钥的每个实例都是唯一的） |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/key\_derivation\_function\_ctx.h" | |
| **Description** | 添加存储在（非秘密）ReadOnlyMemRegion 中的盐值。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00163] AddSecretSalt

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00163 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21513 SWS\_CRYPT\_00609 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | AddSecretSalt(const SecretSeed &salt) | |
| **Scope** | class ara::crypto::cryp::KeyDerivationFunctionCtx | |
| **Syntax** | virtual ara::core::Result<void> AddSecretSalt (const SecretSeed &salt) noexcept=0 | |
| **Parameters (in)** | salt | 一个盐值（如果使用，它应该对于目标密钥的每个实例都是唯一的） |
| **Return value** | ara::core::Result< void > |  |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/key\_derivation\_function\_ctx.h" | |
| **Description** | 添加存储在 SecretSeed 对象中的秘密盐值 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-00164] ConfigIterations

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00164 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21514 SWS\_CRYPT\_00610 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ConfigIterations(std::uint32\_t iterations=0) | |
| **Scope** | class ara::crypto::cryp::KeyDerivationFunctionCtx | |
| **Syntax** | virtual std::uint32\_t ConfigIterations (std::uint32\_t iterations=0) noexcept=0; | |
| **Parameters (in)** | iterations | 基本函数所需的迭代次数（0 表示实现默认次数） |
| **Return value** | std::uint32\_t | 现在在上下文中配置的实际迭代次数（在此方法调用之后） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/key\_derivation\_function\_ctx.h" | |
| **Description** | 配置默认情况下将应用的迭代次数。 实现可以限制迭代次数的最小值和/或最大值。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00165] DeriveKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00165 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21515 SWS\_CRYPT\_00611 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | DeriveKey(bool isSession=true, bool isExportable=false) | |
| **Scope** | class ara::crypto::cryp::KeyDerivationFunctionCtx | |
| **Syntax** | virtual ara::core::Result<SymmetricKey::Uptrc> DeriveKey (bool is Session=true, bool isExportable=false) const noexcept=0; | |
| **Parameters (in)** | isSession | 目标密钥的“会话”（或“临时”）属性（如果为true） |
| isExportable | 目标密钥的可导出性属性（如果为true） |
| **Return value** | ara::core::Result<SymmetricKey::Uptrc> | 指向已创建派生对称密钥实例的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUninitialized Context | 如果上下文没有充分初始化 |
| **Header file** | #include "ara/crypto/cryp/key\_derivation\_function\_ctx.h" | |
| **Description** | 从提供的密钥材料和提供的上下文配置派生对称密钥 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00166] DeriveSeed

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00166 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21516 SWS\_CRYPT\_00611 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | DeriveSeed(bool isSession=true, bool isExportable=false) | |
| **Scope** | class ara::crypto::cryp::KeyDerivationFunctionCtx | |
| **Syntax** | virtual ara::core::Result<SecretSeed::Uptrc> DeriveSeed (bool is Session=true, bool isExportable=false) const noexcept=0; | |
| **Parameters (in)** | isSession | 目标密钥的“会话”（或“临时”）属性（如果为true） |
| isExportable | 目标密钥的可导出性属性（如果为true） |
| **Return value** | ara::core::Result<SecretSeed::Uptrc> | 指向已创建派生秘密种子实例的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUninitialized Context | 如果上下文没有充分初始化 |
| **Header file** | #include "ara/crypto/cryp/key\_derivation\_function\_ctx.h" | |
| **Description** | 从提供的“主”密钥材料和提供的上下文配置派生“从”密钥材料（秘密种子） | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00167] Reset

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00167 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21524 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Reset() | |
| **Scope** | class ara::crypto::cryp::KeyDerivationFunctionCtx | |
| **Syntax** | virtual ara::core::Result<void> Reset () noexcept=0; | |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/key\_derivation\_function\_ctx.h" | |
| **Description** | 清除加密上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00168] GetExtensionService

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00168 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21517 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetExtensionService() | |
| **Scope** | class ara::crypto::cryp::KeyDerivationFunctionCtx | |
| **Syntax** | virtual ExtensionService::Uptr GetExtensionService () const noexcept=0; | |
| **Return value** | ExtensionService::Uptr | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/key\_derivation\_function\_ctx.h" | |
| **Description** | 获取 ExtensionService 实例。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00169] GetKeyIdSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00169 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21519 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetKeyIdSize() | |
| **Scope** | class ara::crypto::cryp::KeyDerivationFunctionCtx | |
| **Syntax** | virtual std::size\_t GetKeyIdSize () const noexcept=0; | |
| **Return value** | std::size\_t | 密钥 ID 的大小（以字节为单位） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/key\_derivation\_function\_ctx.h" | |
| **Description** | 获取多样化算法所需的目标密钥ID的固定大小。 接口的每个实例的返回值都是常数，即独立于配置。 对于密钥派生上下文的此实例，返回值是常量，即独立于配置 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00170] GetTargetAlgId

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00170 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21520 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetTargetAlgId() | |
| **Scope** | class ara::crypto::cryp::KeyDerivationFunctionCtx | |
| **Syntax** | virtual AlgId GetTargetAlgId () const noexcept=0; | |
| **Return value** | AlgId | 目标密钥的对称算法 ID，由最后一次调用返回的 Init() 方法配置。 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/key\_derivation\_function\_ctx.h" | |
| **Description** | 获取目标（从）密钥的对称算法 ID。 如果尚未通过调用 Init() 方法配置上下文，则应该是kAlgIdUndefined。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00171] GetTargetAllowedUsage

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00171 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21521 SWS\_CRYPT\_40946 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetTargetAllowedUsage() | |
| **Scope** | class ara::crypto::cryp::KeyDerivationFunctionCtx | |
| **Syntax** | virtual AllowedUsageFlags GetTargetAllowedUsage () const noexcept=0; | |
| **Return value** | AllowedUsageFlags | 目标密钥的允许密钥使用位标志 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/key\_derivation\_function\_ctx.h" | |
| **Description** | 获取目标（从）密钥的允许密钥用法。 返回值取决于源密钥材料允许使用标志和 Init() 方法最后一次调用的参数 allowedUsage。 如果尚未通过调用 Init() 方法配置上下文，则应返回源密钥材料的允许使用标志。 如果尚未通过调用 Init() 方法配置上下文并且也未设置源密钥材料，则应返回 kAllow KdfMaterialAnyUsage。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00172] GetTargetKeyBitLength

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00172 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21522 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetTargetKeyBitLength() | |
| **Scope** | class ara::crypto::cryp::KeyDerivationFunctionCtx | |
| **Syntax** | virtual std::size\_t GetTargetKeyBitLength () const noexcept=0; | |
| **Return value** | std::size\_t | 目标密钥的长度（以比特为单位） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/key\_derivation\_function\_ctx.h" | |
| **Description** | 获取目标密钥的位长。 返回值由上下文工厂方法配置，即独立于配置方式。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00173] Init

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00173 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21523 SWS\_CRYPT\_40945 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Init(ReadOnlyMemRegion targetKeyId, AlgId targetAlgId=kAlgIdAny, AllowedUsageFlags allowedUsage=kAllowKdfMaterialAnyUsage, ReadOnlyMemRegion ctxLabel=ReadOnlyMem Region()) | |
| **Scope** | class ara::crypto::cryp::KeyDerivationFunctionCtx | |
| **Syntax** | virtual ara::core::Result<void> Init (ReadOnlyMemRegion targetKeyId, AlgId targetAlgId=kAlgIdAny, AllowedUsageFlags allowedUsage=kAllowKdfMaterialAnyUsage, ReadOnlyMemRegion ctxLabel=ReadOnlyMemRegion()) noexcept=0; | |
| **Parameters (in)** | targetKeyId | 目标密钥的ID |
| targetAlgId | 目标对称加密算法标识符 |
| allowedUsage | 定义允许转换类型列表的位标志，其中可以使用目标密钥 |
| ctxLabel | 一个可选的特定于应用程序的“上下文标签”（这可以识别目标密钥和/或通信方的目的） |
| **Return value** | ara::core::Result<void> | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncompatible Arguments | 如果 targetAlgId 指定了一种不同于对称算法的加密算法，其密钥长度等于 GetTargetKeyBitLength()； |
| CryptoErrorDomain::kUsageViolation | 如果 allowedUsage 指定派生密钥材料的使用次数多于源密钥材料，即派生密钥材料的使用不能扩展到源密钥材料允许的范围之外 |
| **Header file** | #include "ara/crypto/cryp/key\_derivation\_function\_ctx.h" | |
| **Description** | 通过最后设置目标密钥 ID 来初始化此上下文。 通过参数 ctxLabel 提供的字节序列可以包含几个不同含义的字段，由单个 0x00 字节分隔。 如果 (targetAlgId == kAlgIdAny) 则可以将派生的密钥加载到支持相同密钥长度的任何对称上下文中（如果还满足“允许使用”标志）！ | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00174] SetSourceKeyMaterial

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00174 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21525 SWS\_CRYPT\_40944 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SetSourceKeyMaterial(const RestrictedUseObject &sourceKM) | |
| **Scope** | class ara::crypto::cryp::KeyDerivationFunctionCtx | |
| **Syntax** | virtual ara::core::Result<void> SetSourceKeyMaterial (const Restricted UseObject &sourceKM) noexcept=0; | |
| **Parameters (in)** | sourceKM | 源密钥材料 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncompatible Object | 如果提供的密钥对象与此对称密钥上下文不兼容 |
| CryptoErrorDomain::kUsageViolation | 如果提供的源密钥材料的“允许使用”限制禁止派生密钥 |
| CryptoErrorDomain::kUsageViolation | 如果源的密钥长度低于内部定义的限制 |
| **Header file** | #include "ara/crypto/cryp/key\_derivation\_function\_ctx.h" | |
| **Description** | 将密钥材料设置（部署）到密钥派生算法上下文。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00175] GetEncapsulatedSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00175 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21818 SWS\_CRYPT\_03008 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetEncapsulatedSize() | |
| **Scope** | class ara::crypto::cryp::KeyEncapsulatorPublicCtx | |
| **Syntax** | virtual std::size\_t GetEncapsulatedSize () const noexcept=0; | |
| **Return value** | std::size\_t | 封装数据块的大小（以字节为单位） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/key\_encapsulator\_public\_ctx.h" | |
| **Description** | 获取封装数据块的固定大小。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00176] GetExtensionService

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00176 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21802 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetExtensionService() | |
| **Scope** | class ara::crypto::cryp::KeyEncapsulatorPublicCtx | |
| **Syntax** | virtual ExtensionService::Uptr GetExtensionService () const noexcept=0; | |
| **Return value** | ExtensionService::Uptr | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/key\_encapsulator\_public\_ctx.h" | |
| **Description** | 获取 ExtensionService 实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00177] GetKekEntropy

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00177 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21817 SWS\_CRYPT\_03007 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetKekEntropy() | |
| **Scope** | class ara::crypto::cryp::KeyEncapsulatorPublicCtx | |
| **Syntax** | virtual std::size\_t GetKekEntropy () const noexcept=0 | |
| **Return value** | std::size\_t | KEK 材料的熵（以位为单位） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/key\_encapsulator\_public\_ctx.h" | |
| **Description** | 获取密钥加密密钥 (KEK) 材料的熵（位长）。 对于 RSA 系统，返回值对应于模 N 的长度（减 1）。 对于 DH 类系统，返回值对应于模 q 的长度（减 1）。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00178] AddKeyingData

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00178 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21810 SWS\_CRYPT\_03000 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | AddKeyingData(const RestrictedUseObject &keyingData) | |
| **Scope** | class ara::crypto::cryp::KeyEncapsulatorPublicCtx | |
| **Syntax** | virtual ara::core::Result<void> AddKeyingData (const RestrictedUse Object &keyingData) noexcept=0; | |
| **Parameters (in)** | keyingData | 要保护的有效载荷 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrc::kUsageViolation | 如果由于 CryptoObject::IsExportable() 返回 FALSE 而无法导出 keyingData |
| CryptoErrc::kIncompatibleObject | 如果 keyingData 属于不同的 Crypto Provider |
| CryptoErrc::kInvalidInputSize | 如果此上下文不支持keyingData的大小 |
| **Header file** | #include "ara/crypto/cryp/key\_encapsulator\_public\_ctx.h" | |
| **Description** | 根据 RFC 5990（“密钥数据”）添加要封装的内容（有效负载）。 目前仅支持 SymmetricKey 和 SecretSeed 对象。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00179] Encapsulate

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00179 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21813 SWS\_CRYPT\_03002 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Encapsulate(KeyDerivationFunctionCtx &kdf, CryptoAlgId kekAlgId) | |
| **Scope** | class ara::crypto::cryp::KeyEncapsulatorPublicCtx | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > Encapsulate (KeyDerivationFunctionCtx &kdf, CryptoAlgId kekAlgId) const noexcept=0; | |
| **Parameters (in)** | kdf | 密钥派生函数的上下文，应用于目标 KEK 生产 |
|  | kekAlgId | 目标 KEK 的算法 ID |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 封装的数据作为字节向量 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrc::kUninitializedContex | 如果上下文不是由公钥 valueErrors 初始化的 |
| CryptoErrc::kInvalidArgument | 如果 kekAlgId 或 kdf 与此上下文不兼容 |
| **Header file** | #include "ara/crypto/cryp/key\_encapsulator\_public\_ctx.h" | |
| **Description** | 封装最后设置的keying-data。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00180] Reset

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00180 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21816 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Reset() | |
| **Scope** | class ara::crypto::cryp::KeyEncapsulatorPublicCtx | |
| **Syntax** | virtual ara::core::Result<void> Reset () noexcept=0; | |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/key\_encapsulator\_public\_ctx.h" | |
| **Description** | 清除加密上下文； | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00181] SetKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00181 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_21815 SWS\_CRYPT\_03006 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SetKey(const PublicKey &key) | |
| **Scope** | class ara::crypto::cryp::KeyEncapsulatorPublicCtx | |
| **Syntax** | virtual ara::core::Result<void> SetKey (const PublicKey &key) noexcept=0; | |
| **Parameters (in)** | key | 源密钥对象 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrc::kIncompatibleObject | 如果提供的密钥对象与此对称密钥上下文不兼容 |
| CryptoErrc::kUsageViolation | 如果与此上下文关联的转换类型被提供的密钥对象的“允许使用”限制禁止 |
| **Header file** | #include "ara/crypto/cryp/key\_encapsulator\_public\_ctx.h" | |
| **Description** | 设置（部署）密钥封装器公共算法上下文的密钥。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00182] Check

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00182 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22119 SWS\_CRYPT\_01213 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Check(const Signature &expected) | |
| **Scope** | class ara::crypto::cryp::MessageAuthnCodeCtx | |
| **Syntax** | virtual ara::core::Result<bool> Check (const Signature &expected) const noexcept=0; | |
| **Parameters (in)** | expected | 包含预期摘要值的签名对象 |
| **Return value** | ara::core::Result< bool > | 如果提供的“签名”对象的值和元信息分别与计算的摘要和上下文的当前配置相同，则为 true； 但否则为false |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Finished | 如果未通过调用 Finish() 方法完成摘要计算 |
| CryptoErrorDomain::kIncompatible Object | 如果提供的“签名”对象是由另一个加密原语类型产生的 |
| **Header file** | #include "ara/crypto/cryp/message\_authn\_code\_ctx.h" | |
| **Description** | 根据预期的“签名”对象检查计算的摘要。 整个摘要值保存在上下文中，直到下次调用 Start()，因此可以再次验证或提取它。 此方法可以在函数 ara::core::memcmp() 标准化后实现为“内联”。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00183] Finish

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00183 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22115 SWS\_CRYPT\_01207 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Finish(bool makeSignatureObject=false) | |
| **Scope** | class ara::crypto::cryp::MessageAuthnCodeCtx | |
| **Syntax** | virtual ara::core::Result<Signature::Uptrc> Finish (bool makeSignature Object=false) noexcept=0; | |
| **Parameters (in)** | makeSignatureObject | 如果此参数为true，则该方法还将生成签名对象 |
| **Return value** | ara::core::Result< Signature::Uptrc > | 指向创建的签名对象的唯一智能指针，如果 (makeSignatureObject == true) 或 nullptr 如果 (make SignatureObject == false) |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Started | 如果摘要计算不是通过调用 Start() 方法启动的 |
| CryptoErrorDomain::kUsageViolatio | 如果缓冲的摘要属于由没有 kAllowSignature权限的密钥初始化的 MAC/HMAC/AE/AEAD 上下文，但是 (makeSignatureObject == true) |
| **Header file** | #include "ara/crypto/cryp/message\_authn\_code\_ctx.h" | |
| **Description** | 完成摘要计算并可选择生成“签名”对象。 只有调用此方法后，才能对摘要进行签名、验证、提取或比较！ 如果签名对象由密钥 MAC/HMAC/AE/AEAD 算法生成，则“签名”的依赖 COUID 应设置为使用的对称密钥的 COUID。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00184] GetDigestService

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00184 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22102 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetDigestService() | |
| **Scope** | class ara::crypto::cryp::MessageAuthnCodeCtx | |
| **Syntax** | virtual DigestService::Uptr GetDigestService () const noexcept=0; | |
| **Return value** | DigestService::Uptr | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/message\_authn\_code\_ctx.h" | |
| **Description** | 获取 DigestService 实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00185] GetDigest

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00185 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22116 SWS\_CRYPT\_01210 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetDigest(std::size\_t offset=0) | |
| **Scope** | class ara::crypto::cryp::MessageAuthnCodeCtx | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > Get Digest (std::size\_t offset=0) const noexcept=0; | |
| **Parameters (in)** | offse | 应该放置到输出缓冲区的摘要的第一个字节的位置 |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 实际存储到输出缓冲区的摘要字节数（它们总是 <= output.size() 并在下面表示为 return\_size） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Finished | 如果未通过调用 Finish() 方法完成摘要计算 |
| CryptoErrorDomain::kUsageViolation | 如果缓冲的摘要属于由没有 kAllowSignature权限的密钥初始化的 MAC/HMAC/AE/AEAD 上下文 |
| **Header file** | #include "ara/crypto/cryp/message\_authn\_code\_ctx.h" | |
| **Description** | 将计算摘要的请求部分获取到现有内存缓冲区。 整个摘要值保存在上下文中直到下一次调用 Start()，因此它的任何部分都可以再次提取或验证。 如果 (full\_digest\_size <= offset) 那么 return\_size = 0 字节； 否则 return\_size = min(output.size(), (full\_digest\_size - offset)) 字节。 此方法可以在函数 ara::core::memcpy() 标准化后实现为“内联”。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00186] GetDigest

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00186 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22117 SWS\_CRYPT\_01210 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetDigest(std::size\_t offset=0) | |
| **Scope** | class ara::crypto::cryp::MessageAuthnCodeCtx | |
| **Syntax** | template <typename Alloc = <implementation-defined>> ara::core::Result<ByteVector<Alloc> > GetDigest (std::size\_t offset=0) const noexcept; | |
| **Template param** | Alloc | 输出容器的自定义分配器类型 |
| **Parameters (in)** | offset | 应该放置到输出缓冲区的摘要的第一个字节的位置 |
| **Return value** | ara::core::Result< ByteVector< Alloc > > |  |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Finished | 如果未通过调用 Finish() 方法完成摘要计算 |
| CryptoErrorDomain::kUsageViolation | 如果缓冲的摘要属于由没有 kAllowSignature权限的密钥初始化的 MAC/HMAC/AE/AEAD 上下文 |
| **Header file** | #include "ara/crypto/cryp/message\_authn\_code\_ctx.h" | |
| **Description** | 将计算摘要的请求部分获取到预先保留的托管容器。 该方法根据实际保存的值设置输出容器的大小。 整个摘要值保存在上下文中直到下一次调用 Start()，因此它的任何部分都可以再次提取或验证。 如果 (full\_digest\_size <= offset) 那么 return\_size = 0 字节； 否则 return\_size = min(output.capacity(), (full\_digest\_size - offset)) 字节。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00187] Reset

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00187 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22120 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Reset() | |
| **Scope** | class ara::crypto::cryp::MessageAuthnCodeCtx | |
| **Syntax** | virtual ara::core::Result<void> Reset () noexcept=0; | |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/message\_authn\_code\_ctx.h" | |
| **Description** | 清除加密上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00188] SetKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00180 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22118 SWS\_CRYPT\_01211 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SetKey(const SymmetricKey &key, CryptoTransform transform=CryptoTransform::kMac Generate) | |
| **Scope** | class ara::crypto::cryp::MessageAuthnCodeCtx | |
| **Syntax** | virtual ara::core::Result<void> SetKey (const SymmetricKey &key, CryptoTransform transform=CryptoTransform::kMacGenerate) noexcept=0; | |
| **Parameters (in)** | key | 源密钥对象 |
| transform | “方向”指标：为直接转换（如果为true）或反向转换（如果为false）部署密钥 |
| **Return value** | ara::core::Result< void > |  |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncompatible Object | 如果提供的密钥对象与此对称密钥上下文不兼容 |
| CryptoErrorDomain::kUsageViolation | 如果与此上下文关联的转换类型（考虑到由转换指定的方向）被提供的密钥对象的“允许使用”限制禁止 |
| **Header file** | #include "ara/crypto/cryp/message\_authn\_code\_ctx.h" | |
| **Description** | 设置（部署）消息验证码算法上下文的密钥。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00189] Start

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00189 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22110 SWS\_CRYPT\_01201 SWS\_CRYPT\_01202 SWS\_CRYPT\_01203 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Start(ReadOnlyMemRegion iv=ReadOnlyMemRegion()) | |
| **Scope** | class ara::crypto::cryp::MessageAuthnCodeCtx | |
| **Syntax** | virtual ara::core::Result<void> Start (ReadOnlyMemRegion iv=ReadOnly MemRegion()) noexcept=0; | |
| **Parameters (in)** | iv | 可选的初始化向量 (IV) 或“nonce”值 |
| **Return value** | ara::core::Result< void > |  |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUninitialized Context | 如果未通过部署密钥初始化上下文 |
| CryptoErrorDomain::kInvalidInputSize | 如果不支持提供的 IV 的大小（即如果它不足以进行初始化） |
| CryptoErrorDomain::kUnsupported | 如果基本算法（或其当前实现）主要不支持 IV 变量，但提供的 IV 值不为空，即 if (iv.empty() == false) |
| **Header file** | #include "ara/crypto/cryp/message\_authn\_code\_ctx.h" | |
| **Description** | 为新的数据流处理或生成初始化上下文（取决于原语）。 如果 IV 大小大于算法支持的最大值，则实现可以仅使用序列中的前导字节 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00190] Start

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00190 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22111 SWS\_CRYPT\_01201 SWS\_CRYPT\_01202 SWS\_CRYPT\_01203 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Start(const SecretSeed &iv) | |
| **Scope** | class ara::crypto::cryp::MessageAuthnCodeCtx | |
| **Syntax** | virtual ara::core::Result<void> Start (const SecretSeed &iv) noexcept=0; | |
| **Parameters (in)** | iv | 初始化向量 (IV) 或“nonce”对象 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUninitialized Context | 如果未通过部署密钥初始化上下文 |
| CryptoErrorDomain::kInvalidInputSize | 如果不支持提供的 IV 的大小（即如果它不足以进行初始化） |
| CryptoErrorDomain::kUnsupported | 如果基本算法（或其当前实现）主要不支持 IV 变量 |
| CryptoErrorDomain::kUsageViolation | 如果提供的 Secret Seed 对象的“允许使用”限制禁止此转换类型 |
| **Header file** | #include "ara/crypto/cryp/message\_authn\_code\_ctx.h" | |
| **Description** | 为新的数据流处理或生成初始化上下文（取决于原语）。 如果 IV 大小大于算法支持的最大值，则实现可以仅使用序列中的前导字节 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00191] Update

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00191 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22112 SWS\_CRYPT\_01204 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Update(const RestrictedUseObject &in) | |
| **Scope** | class ara::crypto::cryp::MessageAuthnCodeCtx | |
| **Syntax** | virtual ara::core::Result<void> Update (const RestrictedUseObject &in) noexcept=0; | |
| **Parameters (in)** | in | 应处理的输入消息的一部分 |
| **Return value** | ara::core::Result< void > |  |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Started | 如果摘要计算不是通过调用 Start() 方法启动的 |
| **Header file** | #include "ara/crypto/cryp/message\_authn\_code\_ctx.h" | |
| **Description** | 通过消息的新部分更新摘要计算上下文。 此方法专用于 RestrictedUseObject 是“消息”的一部分的情况。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00192] Update

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00192 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22113 SWS\_CRYPT\_01204 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Update(ReadOnlyMemRegion in) | |
| **Scope** | class ara::crypto::cryp::MessageAuthnCodeCtx | |
| **Syntax** | virtual ara::core::Result<void> Update (ReadOnlyMemRegion in) noexcept=0; | |
| **Parameters (in)** | in | 应处理的输入消息的一部分 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Started | 如果摘要计算不是通过调用 Start() 方法启动的 |
| **Header file** | #include "ara/crypto/cryp/message\_authn\_code\_ctx.h" | |
| **Description** | 通过消息的新部分更新摘要计算上下文。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00193] Update

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00193 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22114 SWS\_CRYPT\_01204 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Update(std::uint8\_t in) | |
| **Scope** | class ara::crypto::cryp::MessageAuthnCodeCtx | |
| **Syntax** | virtual ara::core::Result<void> Update (std::uint8\_t in) noexcept=0; | |
| **Parameters (in)** | in | 作为输入消息一部分的字节值 |
| **Return value** | ara::core::Result< void > |  |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kProcessingNot Started | 如果摘要计算不是通过调用 Start() 方法启动的 |
| **Header file** | #include "ara/crypto/cryp/message\_authn\_code\_ctx.h" | |
| **Description** | 通过消息的新部分更新摘要计算上下文。 这种方法便于常量标签的处理。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00194] GetExtensionService

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00194 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22210 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetExtensionService() | |
| **Scope** | class ara::crypto::cryp::MsgRecoveryPublicCtx | |
| **Syntax** | virtual ExtensionService::Uptr GetExtensionService () const noexcept=0; | |
| **Return value** | ExtensionService::Uptr | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/msg\_recovery\_public\_ctx.h" | |
| **Description** | 获取 ExtensionService 实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00195] GetMaxInputSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00195 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22213 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetMaxInputSize(bool suppressPadding=false) | |
| **Scope** | class ara::crypto::cryp::MsgRecoveryPublicCtx | |
| **Syntax** | virtual std::size\_t GetMaxInputSize (bool suppressPadding=false) const noexcept=0; | |
| **Parameters (in)** | suppressPadding | 如果为真，则该方法计算纯数据块的整个空间仅用于负载时的大小 |
| **Return value** | std::size\_ | 输入数据块的最大大小，以字节为单位 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/msg\_recovery\_public\_ctx.h" | |
| **Description** | 获取输入数据块的最大期望大小。如果(IsEncryption() == false)，则该方法返回的值独立于suppressPadding参数，且该值将等于块大小。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00196] GetMaxOutputSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00196 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22214 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetMaxOutputSize(bool suppressPadding=false) | |
| **Scope** | class ara::crypto::cryp::MsgRecoveryPublicCtx | |
| **Syntax** | virtual std::size\_t GetMaxOutputSize (bool suppressPadding=false) const noexcept=0 | |
| **Parameters (in)** | suppressPadding | 如果为真，则该方法计算纯数据块的整个空间仅用于负载时的大小 |
| **Return value** | std::size\_t | 输出数据块的最大大小，以字节为单位 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/msg\_recovery\_public\_ctx.h" | |
| **Description** | 获取输出数据块的最大可能大小。如果(IsEncryption() == true)，则该方法返回的值独立于suppressPadding参数，并将等于块大小。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00197] DecodeAndVerify

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-ID (SWRD-ID 编号规则参见附录A-信息定义需求ID。) | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22215 SWS\_CRYPT\_02421 SWS\_CRYPT\_02422 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | DecodeAndVerify(ReadOnlyMemRegion in) | |
| **Scope** | class ara::crypto::cryp::MsgRecoveryPublicCtx | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > Decode AndVerify (ReadOnlyMemRegion in) const noexcept=0; | |
| **Parameters (in)** | in | 输入数据块 |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 输出数据的输出缓冲区实际大小(它总是<= out.size())，如果输入数据块有错误的内容，则为0 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncorrectInput Size | 如果违反了上面提到的关于输入大小的规则 |
| CryptoErrorDomain::kUninitialized Context | 如果上下文没有被密钥值初始化 |
| **Header file** | #include "ara/crypto/cryp/msg\_recovery\_public\_ctx.h" | |
| **Description** | 根据加密配置处理(加密/解密)一个输入块。加密(suppressPadding == true)期望:in.size() == GetMaxOutputSize(true) && out.size() >= GetMaxOutputSize(true)。加密(suppressPadding == false)期望:in.size() <= GetMaxOutputSize(false) && in.size() > 0 && out.size() >= GetMaxOutputSize(false)。解密期望:in.size() == GetMaxInputSize() && out.size() >= GetMaxOutput Size(suppressPadding)。在 (out.size() < GetMaxOutputSize())情况下应该谨慎使用，除非你严格确定输出数据的大小!在这种情况下(suppressPadding == true)纯文本的实际大小应该等于纯数据块的完整大小(由算法定义)! | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00198] DecodeAndVerify

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00198 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22216 SWS\_CRYPT\_02421 SWS\_CRYPT\_02422 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | DecodeAndVerify(ReadOnlyMemRegion in) | |
| **Scope** | class ara::crypto::cryp::MsgRecoveryPublicCtx | |
| **Syntax** | template <typename Alloc = <implementation-defined>> ara::core::Result<ByteVector<Alloc> > DecodeAndVerify (ReadOnlyMem Region in) const noexcept; | |
| **Template param** | Alloc | 输出容器的自定义分配器类型 |
| **Parameters (in)** | in | 输入数据块 |
| **Return value** | ara::core::Result< ByteVector< Alloc > > | 输出块的托管容器 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncorrectInput Size | 如果违反了上面提到的关于输入大小的规则 |
| CryptoErrorDomain::kUninitialized Context | 如果上下文没有被密钥值初始化 |
| **Header file** | #include "ara/crypto/cryp/msg\_recovery\_public\_ctx.h" | |
| **Description** | 根据加密配置处理(加密/解密)一个输入块。这个方法根据实际保存的值来设置输出容器的大小!加密与(suppressPadding == true)期望:in.size() == GetMaxOutputSize(true) && out.capacity() >= GetMaxOutputSize(true)。加密(suppressPadding == false)期望:in.size() <= GetMaxInputSize(false) && in.size() > 0 && out.capacity() >= GetMaxOutput Size(false)。解密期望:in.size() == getmaxoutputsize () && out.capacity() >= Get MaxOutputSize(suppressPadding)。在(out.capacity() < GetMaxOutputSize())情况下应该谨慎使用，除非您严格确定输出数据的大小!在(suppressPadding == true)的情况下，纯文本的实际大小应该等于纯数据块(由算法定义)的完整大小! | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00199] Reset

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00199 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22212 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Reset() | |
| **Scope** | class ara::crypto::cryp::MsgRecoveryPublicCtx | |
| **Syntax** | virtual ara::core::Result<void> Reset () noexcept=0; | |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/msg\_recovery\_public\_ctx.h" | |
| **Description** | 清除上下文 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-00200] SetKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00200 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22211 SWS\_CRYPT\_01823 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SetKey(const PublicKey &key) | |
| **Scope** | class ara::crypto::cryp::MsgRecoveryPublicCtx | |
| **Syntax** | virtual ara::core::Result<void> SetKey (const PublicKey &key) noexcept=0; | |
| **Parameters (in)** | key | 源密钥对象 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncompatible Object | 如果提供的密钥对象与该对称密钥上下文不兼容 |
| CryptoErrorDomain::kUsageViolation | 如果与此上下文关联的转换类型被所提供的键对象的“允许使用”限制所禁止 |
| **Header file** | #include "ara/crypto/cryp/msg\_recovery\_public\_ctx.h" | |
| **Description** | 为消息恢复公共算法上下文设置(deploy)一个密钥。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00201] GetPublicKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00201 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22511 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetPublicKey() | |
| **Scope** | class ara::crypto::cryp::PrivateKey | |
| **Syntax** | virtual ara::core::Result<PublicKey::Uptrc> GetPublicKey () const noexcept=0; | |
| **Return value** | ara::core::Result< PublicKey::Uptrc > | 向与此私钥对应的公钥的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/cryobj/private\_key.h" | |
| **Description** | 获取与此私钥对应的公钥。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00202] CheckKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00202 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22711 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CheckKey(bool strongCheck=true) | |
| **Scope** | class ara::crypto::cryp:: PublicKey | |
| **Syntax** | virtual bool CheckKey (bool strongCheck=true) const noexcept=0; | |
| **Parameters (in)** | strongCheck | 表示所需检查类型的严重性标志:strong(如果为true)或fast(如果为false) |
| **Return value** | bool | 如果密钥是正确的，则为true |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/cryobj/ public\_key.h " | |
| **Description** | 检查密钥的正确性 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-00203] HashPublicKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00203 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22712 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | HashPublicKey(HashFunctionCtx &hashFunc) | |
| **Scope** | class ara::crypto::cryp::PublicKey | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > Hash PublicKey (HashFunctionCtx &hashFunc) const noexcept=0; | |
| **Parameters (in)** | hashFunc | 用于哈希计算的哈希上下文实例 |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 预分配给结果哈希值的缓冲区 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInsufficient Capacity | 如果哈希缓冲区的大小不足以存储结果 |
| CryptoErrorDomain::kIncompleteArg State | 如果hashFunc上下文没有初始化 |
| **Header file** | #include "ara/crypto/cryp/cryobj/ public\_key.h " | |
| **Description** | 计算公钥值的哈希值。原始的公钥值BLOB可以通过Serializable接口获得； | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00204] HashPublicKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00204 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22713 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | HashPublicKey(HashFunctionCtx &hashFunc) | |
| **Scope** | class ara::crypto::cryp::PublicKey | |
| **Syntax** | template <typename Alloc = <implementation-defined>> ara::core::Result<ByteVector<Alloc> > HashPublicKey (HashFunctionCtx &hashFunc) const noexcept; | |
| **Template param** | Alloc | 输出容器的自定义分配器类型 |
| **Parameters (in)** | hashFunc | 用于哈希计算的哈希上下文实例 |
| **Return value** | ara::core::Result< ByteVector< Alloc > > | 为结果散列值预留的托管容器 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInsufficient Capacity | 如果哈希缓冲区的容量不足以存储结果 |
| CryptoErrorDomain::kIncompleteArg State | 如果hashFunc上下文没有初始化 |
| **Header file** | #include "ara/crypto/cryp/cryobj/ public\_key.h " | |
| **Description** | 计算公钥值的哈希值。这个方法根据实际保存的值来设置输出容器的大小!原始的公钥值BLOB可以通过Serializable接口获得。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00205] AddEntropy

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00205 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22914 SWS\_CRYPT\_00502 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | AddEntropy(ReadOnlyMemRegion entropy) | |
| **Scope** | class ara::crypto::cryp::RandomGeneratorCtx | |
| **Syntax** | virtual bool AddEntropy (ReadOnlyMemRegion entropy) noexcept=0; | |
| **Parameters (in)** | entropy | 具有附加熵值的存储区域 |
| **Return value** | bool | 如果支持该方法且熵已成功更新，则为true |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/random\_generator\_ctx.h" | |
| **Description** | 通过将其与提供的附加熵混合，更新RNG的内部状态。此方法对于实现是可选的。这种方法的实现可以“累积”为将来使用提供熵。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00206] Generate

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00206 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22915 SWS\_CRYPT\_00507 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Generate(std::uint32\_t count) | |
| **Scope** | class ara::crypto::cryp::RandomGeneratorCtx | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > Generate (std::uint32\_t count) noexcept=0; | |
| **Parameters (in)** | count | 生成的随机字节数 |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 用生成的随机序列填充的缓冲区 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUninitialized Context | 如果这个上下文实现了一个本地RNG(也就是说，RNG的状态是由应用程序控制的)，并且必须由应用程序进行种子化，因为它要么还没有被种子化，要么已经耗尽了熵。 |
| CryptoErrorDomain::kBusyResource | 如果这个上下文实现了一个全局的RNG(即，RNG的状态是由堆栈控制的，而不是应用程序)，那么这个RNG当前处于out- entropy状态，因此不能提供请求的随机字节数 |
| **Header file** | #include "ara/crypto/cryp/random\_generator\_ctx.h" | |
| **Description** | 返回一个已分配的缓冲区，其中包含一个生成的与请求大小相同的随机序列 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00207] GetExtensionService

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00207 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22902 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetExtensionService() | |
| **Scope** | class ara::crypto::cryp::RandomGeneratorCtx | |
| **Syntax** | virtual ExtensionService::Uptr GetExtensionService () const noexcept=0; | |
| **Return value** | ExtensionService::Uptr | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/random\_generator\_ctx.h" | |
| **Description** | 获取ExtensionService 实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00208] Seed

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00208 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22911 SWS\_CRYPT\_00502 SWS\_CRYPT\_00503 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Seed(ReadOnlyMemRegion seed) | |
| **Scope** | class ara::crypto::cryp::RandomGeneratorCtx | |
| **Syntax** | virtual bool Seed (ReadOnlyMemRegion seed) noexcept=0; | |
| **Parameters (in)** | seed | 带有种子值的内存区域 |
| **Return value** | bool | 如果支持该方法且状态已成功设置，则为true |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/random\_generator\_ctx.h" | |
| **Description** | 使用提供的种子设置RNG的内部状态。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00209] Seed

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00209 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22912 SWS\_CRYPT\_00502 SWS\_CRYPT\_00503 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Seed(const SecretSeed &seed) | |
| **Scope** | class ara::crypto::cryp::RandomGeneratorCtx | |
| **Syntax** | virtual bool Seed (const SecretSeed &seed) noexcept=0; | |
| **Parameters (in)** | seed | 带有种子值的内存区域 |
| **Return value** | bool | 如果支持该方法且状态已成功设置，则为true |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/random\_generator\_ctx.h" | |
| **Description** | 使用提供的种子设置RNG的内部状态。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00210] SetKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00210 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_22913 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SetKey(const SymmetricKey &key) | |
| **Scope** | class ara::crypto::cryp::RandomGeneratorCtx | |
| **Syntax** | virtual bool SetKey (const SymmetricKey &key) noexcept=0; | |
| **Parameters (in)** | key | 一个SymmetricKey，该key用作种子值 |
| **Return value** | bool | 如果支持该方法且密钥已成功设置，则为true |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/random\_generator\_ctx.h" | |
| **Description** | 使用提供的种子设置RNG的内部状态。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00211] GetAllowedUsage

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00211 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24811 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetAllowedUsage() | |
| **Scope** | class ara::crypto::cryp::RestrictedUseObject | |
| **Syntax** | virtual Usage GetAllowedUsage () const noexcept=0; | |
| **Return value** | Usage | 指定对象允许的应用程序的位标志组合 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/cryobj/restricted\_use\_object.h" | |
| **Description** | 获取此对象的允许使用 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00212] Clone

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00212 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23011 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Clone(ReadOnlyMemRegion xorDelta=ReadOnlyMemRegion()) | |
| **Scope** | class ara::crypto::cryp::SecretSeed | |
| **Syntax** | virtual ara::core::Result<SecretSeed::Uptr> Clone (ReadOnlyMemRegion xorDelta=ReadOnlyMemRegion()) const noexcept=0; | |
| **Parameters (in)** | xorDelta | 可选的“delta”值，必须与原始种子的“克隆”副本异或删除 |
| **Return value** | ara::core::Result< SecretSeed::Uptr > | 指向“克隆的”会话Secret Seed对象的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/cryobj/secret\_seed.h" | |
| **Description** | 克隆这个Secret Seed对象到新的会话对象。创建的对象实例是会话和不可导出的，“克隆”对象的AllowedUsageFlags属性与源对象的这个属性相同!如果xorDelta参数的大小小于此种子的值大小，则只有原始种子的相应前导字节数应XOR-ed，但其余的应复制而不更改。如果xorDelta参数的大小大于该种子的值大小，则xorDelta的额外字节应被忽略 | |
| **Additional** | SWRD-API-Crypto-00212 | |

#### [SWRD-API-Crypto-00213] JumpFrom

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00213 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23012 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | JumpFrom(const SecretSeed &from, std::int64\_t steps) | |
| **Scope** | class ara::crypto::cryp::SecretSeed | |
| **Syntax** | virtual ara::core::Result<void> JumpFrom (const SecretSeed &from, std::int64\_t steps) noexcept=0; | |
| **Parameters (in)** | from | 源对象，该对象保持初始值以便从中跳转 |
|  | steps | “跳跃”的步骤数 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncompatible Object | 如果此对象和from参数与不兼容的密码算法相关联 |
| CryptoErrorDomain::kInvalidInputSize | 如果from种子的size值小于这个种子的size值 |
| **Header file** | #include "ara/crypto/cryp/cryobj/secret\_seed.h" | |
| **Description** | 根据与该对象相关的加密算法定义的“计数”表达式，将该种子对象的值设置为从初始状态“跳转”到指定的步骤数。步骤可能有正值和负值，分别对应于“跳转”的前进和后退方向，但0值表示仅从值复制到该种子对象。from参数的种子大小必须大于或等于该种子大小。 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-00214] Jump

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00214 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23014 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Jump(std::int64\_t steps) | |
| **Scope** | class ara::crypto::cryp::SecretSeed | |
| **Syntax** | virtual SecretSeed& Jump (std::int64\_t steps) noexcept=0; | |
| **Parameters (in)** | steps | 从当前状态跳转(向前或向后)的“步骤”数 |
| **Return value** | SecretSeed & | 更新的对象的引用 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/cryobj/secret\_seed.h" | |
| **Description** | 根据与该对象相关的加密算法定义的“计数”表达式，将该种子对象的值设置为从其当前状态“跳转”到指定的步骤数。步骤可能有正值和负值，分别对应于“跳转”的前进和后退方向，但是0值表示当前种子值没有改变。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00215] Next

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00215 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23013 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Next() | |
| **Scope** | class ara::crypto::cryp::SecretSeed | |
| **Syntax** | virtual SecretSeed& Next () noexcept=0; | |
| **Return value** | SecretSeed & | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/cryobj/secret\_seed.h" | |
| **Description** | 根据与此对象相关的密码算法定义的“计数”表达式设置秘密种子的下一个值。如果相关的密码算法没有指定一个“计数”表达式，那么泛型的增量操作必须实现为默认值(小端符号，即第一个字节是最不重要的)。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00216] operatorˆ=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00216 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23015 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operatorˆ=(const SecretSeed &source) | |
| **Scope** | class ara::crypto::cryp::SecretSeed | |
| **Syntax** | virtual SecretSeed& operatorˆ= (const SecretSeed &source) noexcept=0; | |
| **Parameters (in)** | source | 进行XOR操作的右参数 |
| **Return value** | SecretSeed & | 更新的对象引用 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/cryobj/secret\_seed.h" | |
| **Description** | 此种子对象与另一个种子对象的异或值，并将结果保存到此对象。如果该对象和source参数中的种子大小不同，则该种子对象中只有相应的前导字节数应该被更新 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00217] operatorˆ=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00217 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23016 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operatorˆ=(ReadOnlyMemRegion source | |
| **Scope** | class ara::crypto::cryp::SecretSeed | |
| **Syntax** | virtual SecretSeed& operatorˆ= (ReadOnlyMemRegion source) noexcept=0; | |
| **Parameters (in)** | source | 进行XOR操作的右参数 |
| **Return value** | SecretSeed & | 更新的对象引用 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/cryobj/secret\_seed.h" | |
| **Description** | 提供内存区域的种子对象的异或值，并将结果保存到该对象。如果该对象和source参数中的种子大小不同，则该种子对象的前导字节数应该被更新。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00218] CryptoException

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00218 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_19906 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CryptoException(ara::core::ErrorCode err) | |
| **Scope** | class ara::crypto::CryptoException | |
| **Syntax** | explicit CryptoException (ara::core::ErrorCode err) noexcept; | |
| **Parameters (in)** | err | 错误码 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/common/crypto\_error\_domain.h" | |
| **Description** | 从ErrorCode构造一个新的CryptoException。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00219] GetExtensionService

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00219 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23210 | |
| **CR** |  | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetExtensionService() | |
| **Scope** | class ara::crypto::cryp::SigEncodePrivateCtx | |
| **Syntax** | virtual ExtensionService::Uptr GetExtensionService () const noexcept=0; | |
| **Return value** | ExtensionService::Uptr | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/sig\_encode\_private\_ctx.h" | |
| **Description** | 获取ExtensionService实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00220] GetMaxInputSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00220 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23213 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetMaxInputSize(bool suppressPadding=false) | |
| **Scope** | class ara::crypto::cryp::SigEncodePrivateCtx | |
| **Syntax** | virtual std::size\_t GetMaxInputSize (bool suppressPadding=false) const noexcept=0; | |
| **Parameters (in)** | suppressPadding | 如果为真，则该方法计算纯数据块的整个空间仅用于负载时的大小 |
| **Return value** | std::size\_t | 输入数据块的最大大小，以字节为单位 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/sig\_encode\_private\_ctx.h" | |
| **Description** | 获取输入数据块的最大期望大小。如果(IsEncryption() == false)，则该方法返回的值独立于suppressPadding参数，且该值将等于块大小。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00221] GetMaxOutputSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00221 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23214 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetMaxOutputSize(bool suppressPadding=false) | |
| **Scope** | class ara::crypto::cryp::SigEncodePrivateCtx | |
| **Syntax** | virtual std::size\_t GetMaxOutputSize (bool suppressPadding=false) const noexcept=0; | |
| **Parameters (in)** | suppressPadding | 如果为true，则该方法计算纯数据块的整个空间仅用于负载时的大小 |
| **Return value** | std::size\_t | 输出数据块的最大大小，以字节为单位 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/sig\_encode\_private\_ctx.h" | |
| **Description** | 获取输出数据块的最大可能大小。如果(IsEncryption() == true)，则该方法返回的值独立于suppressPadding参数，并将等于块大小。 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-00222] SignAndEncode

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00222 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23215 SWS\_CRYPT\_02420 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SignAndEncode(ReadOnlyMemRegion in) | |
| **Scope** | class ara::crypto::cryp::SigEncodePrivateCtx | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > SignAnd Encode (ReadOnlyMemRegion in) const noexcept=0; | |
| **Parameters (in)** | in | 输入数据块 |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 输出数据的输出缓冲区实际大小(它总是<= out.size())，如果输入数据块有错误的内容，则为0 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncorrectInput Size | 如果违反了上面提到的关于输入大小的规则 |
| CryptoErrorDomain::kUninitialized Context | 如果上下文没有被密钥值初始化 |
| **Header file** | #include "ara/crypto/cryp/sig\_encode\_private\_ctx.h" | |
| **Description** | 根据加密配置处理(加密/解密)一个输入块。加密时(suppressPadding == true)期望:in.size() == GetMaxOutputSize(true) && out.size() >= GetMaxOutputSize(true)。加密时(suppressPadding == false)期望:in.size() <= GetMaxOutputSize(false) && in.size() > 0 && out.size() >= GetMaxOutputSize(false)。解密期望:in.size() == GetMaxInputSize() && out.size() >= GetMaxOutput Size(suppressPadding)。在 (out.size() < GetMaxOutputSize())情况下应该谨慎使用，除非你严格确定输出数据的大小!在这种情况下(suppressPadding == true)纯文本的实际大小应该等于纯数据块的完整大小(由算法定义)! | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00223] SignAndEncode

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00223 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23216 SWS\_CRYPT\_02420 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SignAndEncode(ReadOnlyMemRegion in) | |
| **Scope** | class ara::crypto::cryp::SigEncodePrivateCtx | |
| **Syntax** | template <typename Alloc = <implementation-defined>> ara::core::Result<ByteVector<Alloc> > SignAndEncode (ReadOnlyMemRegion in) const noexcept | |
| **Template param** | Alloc | 输出容器的自定义分配器类型 |
| **Parameters (in)** | in | 输入数据块 |
| **Return value** | ara::core::Result<ByteVector<Alloc> > | 输出块的托管容器 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncorrectInput Size | 如果违反了上面提到的关于输入大小的规则 |
| CryptoErrorDomain::kUninitialized Context | 如果上下文没有被密钥值初始化 |
| **Header file** | #include "ara/crypto/cryp/sig\_encode\_private\_ctx.h" | |
| **Description** | 根据加密配置处理(加密/解密)一个输入块。这个方法根据实际保存的值来设置输出容器的大小!加密时(suppressPadding == true)期望:in.size() == GetMaxOutputSize(true) && out.capacity() >= GetMaxOutputSize(true)。加密时(suppressPadding == false)期望:in.size() <= GetMaxInputSize(false) && in.size() > 0 && out.capacity() >= GetMaxOutput Size(false)。解密期望:in.size() == getmaxoutputsize () && out.capacity() >= Get MaxOutputSize(suppressPadding)。在(out.capacity() < GetMaxOutputSize())情况下应该谨慎使用，除非你严格确定输出数据的大小!在(suppressPadding == true)的情况下，纯文本的实际大小应该等于纯数据块(由算法定义)的完整大小! | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00224] Reset

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00224 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23212 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Reset() | |
| **Scope** | class ara::crypto::cryp::SigEncodePrivateCtx | |
| **Syntax** | virtual ara::core::Result<void> Reset () noexcept=0; | |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/sig\_encode\_private\_ctx.h" | |
| **Description** | 清除加密上下文 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-00225] SetKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00225 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23211 SWS\_CRYPT\_01822 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SetKey(const PrivateKey &key) | |
| **Scope** | class ara::crypto::cryp::SigEncodePrivateCtx | |
| **Syntax** | virtual ara::core::Result<void> SetKey (const PrivateKey &key) noexcept=0; | |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncompatible Object | 如果提供的密钥对象与这个上下文不兼容 |
| CryptoErrorDomain::kUsageViolation | 如果与此上下文关联的转换类型被所提供的密钥对象的“允许使用”限制所禁止 |
| **Header file** | #include "ara/crypto/cryp/sig\_encode\_private\_ctx.h" | |
| **Description** | 设置(部署)一个密钥到签名编码私有算法上下文。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00226] GetHashAlgId

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00226 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23311 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetHashAlgId() | |
| **Scope** | class ara::crypto::cryp::Signature | |
| **Syntax** | virtual CryptoPrimitiveId::AlgId GetHashAlgId () const noexcept=0; | |
| **Return value** | CryptoPrimitiveId::AlgId | 仅使用哈希算法的ID(没有签名算法说明) |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/cryobj/signature.h" | |
| **Description** | 获取用于此签名对象生成的哈希算法的ID。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00227] GetRequiredHashSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00227 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23312 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetRequiredHashSize() | |
| **Scope** | class ara::crypto::cryp::Signature | |
| **Syntax** | virtual std::size\_t GetRequiredHashSize () const noexcept=0; | |
| **Return value** | std::size\_t | 所需散列大小(以字节为单位) |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/cryobj/signature.h" | |
| **Description** | 获取当前签名算法所需的散列大小。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00228] GetRequiredHashAlgId

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00228 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29003 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetRequiredHashAlgId() | |
| **Scope** | class ara::crypto::cryp::SignatureService | |
| **Syntax** | virtual CryptoPrimitiveId::AlgId GetRequiredHashAlgId () const noexcept=0; | |
| **Return value** | CryptoPrimitiveId::AlgId | 需要哈希算法ID或kAlgIdAny，如果签名算法规范没有包含具体的哈希函数 |
| **Exception Safety** | noexcept | |
| **Header file** | #include "ara/crypto/cryp/signature\_service.h" | |
| **Description** | 获取当前签名算法所需的哈希算法的ID | |
| **Additional** |  | |

#### [SWRD-API-Crypto-00229] GetRequiredHashSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00229 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29002 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetRequiredHashSize() | |
| **Scope** | class ara::crypto::cryp::SignatureService | |
| **Syntax** | virtual std::size\_t GetRequiredHashSize () const noexcept=0; | |
| **Return value** | std::size\_t | 所需散列大小(以字节为单位) |
| **Exception Safety** | noexcept | |
| **Header file** | #include "ara/crypto/cryp/signature\_service.h" | |
| **Description** | 获取当前签名算法所需的散列大小。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00230] GetSignatureSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00230 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_29004 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetSignatureSize() | |
| **Scope** | class ara::crypto::cryp::SignatureService | |
| **Syntax** | virtual std::size\_t GetSignatureSize () const noexcept=0; | |
| **Return value** | std::size | 签名值大小(以字节为单位) |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/signature\_service.h" | |
| **Description** | 获取当前算法产生和需要的签名值的大小。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00231] GetSignatureService

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00231 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23510 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetSignatureService() | |
| **Scope** | class ara::crypto::cryp::SignerPrivateCtx | |
| **Syntax** | virtual SignatureService::Uptr GetSignatureService () const noexcept=0 | |
| **Return value** | SignatureService::Uptr | - |
| **Exception Safety** | noexcept | |
| **Header file** | #include "ara/crypto/cryp/signer\_private\_ctx.h" | |
| **Description** | 获取SignatureService实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00232] Reset

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00232 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23516 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Reset() | |
| **Scope** | class ara::crypto::cryp::SignerPrivateCtx | |
| **Syntax** | virtual ara::core::Result<void> Reset () noexcept=0; | |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/signer\_private\_ctx.h" | |
| **Description** | 清除上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00233] SetKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00233 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23515 SWS\_CRYPT\_01820 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SetKey(const PrivateKey &key) | |
| **Scope** | class ara::crypto::cryp::SignerPrivateCtx | |
| **Syntax** | virtual ara::core::Result<void> SetKey (const PrivateKey &key) noexcept=0; | |
| **Parameters (in)** | key | 源密钥对象 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncompatible Object | 如果提供的密钥对象与这个上下文不兼容 |
| CryptoErrorDomain::kUsageViolation | 如果与此上下文关联的转换类型被所提供的密钥对象的“允许使用”限制所禁止 |
| **Header file** | #include "ara/crypto/cryp/signer\_private\_ctx.h" | |
| **Description** | 为签名私钥算法上下文设置(部署)密钥。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00234] SignPreHashed

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00234 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23511 SWS\_CRYPT\_02415 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SignPreHashed(const HashFunctionCtx &hashFn, ReadOnlyMemRegion context=ReadOnly MemRegion()) | |
| **Scope** | class ara::crypto::cryp::SignerPrivateCtx | |
| **Syntax** | virtual ara::core::Result<Signature::Uptrc> SignPreHashed (const Hash FunctionCtx &hashFn, ReadOnlyMemRegion context=ReadOnlyMemRegion()) const noexcept=0; | |
| **Parameters (in)** | hashFn | 哈希功能上下文，其中包含用于签名的摘要值 |
| context | 用户提供的可选“上下文”(它的支持取决于具体的算法) |
| **Return value** | ara::core::Result< Signature::Uptrc > | 指向序列化签名的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgumen | 如果哈希函数算法不符合此上下文的签名算法规范 |
| CryptoErrorDomain::kInvalidInputSize | 如果用户提供的上下文大小不正确(或不支持) |
|  | CryptoErrorDomain::kProcessingNot Finished | 如果方法hash.Finish()在调用此方法之前没有被调用 |
|  | CryptoErrorDomain::kUninitialized Context | 此上下文未由密钥值初始化 |
| **Header file** | #include "ara/crypto/cryp/signer\_private\_ctx.h" | |
| **Description** | 对存储在哈希函数上下文中的提供的摘要值进行签名。此方法必须将所用密钥对的哈希函数算法ID和COUID放到结果签名对象中!用户提供的context可用于以下算法:Ed25519ctx, Ed25519ph, Ed448ph。如果目标算法不支持context参数，则必须提供空值(默认值)! | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00235] Sign

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00235 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23512 SWS\_CRYPT\_02416 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Sign(ReadOnlyMemRegion value, ReadOnlyMemRegion context=ReadOnlyMemRegion()) | |
| **Scope** | class ara::crypto::cryp::SignerPrivateCtx | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > Sign (ReadOnlyMemRegion value, ReadOnlyMemRegion context=ReadOnlyMem Region()) const noexcept=0; | |
| **Parameters (in)** | value | 应签名的(预)散列或直接消息值 |
| context | 用户提供的可选“上下文”(它的支持取决于具体的算法) |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 存储到输出缓冲区的签名值的实际大小 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidInputSize | 如果输入值或上下文参数的大小不正确/不受支持 |
| CryptoErrorDomain::kUninitialized Context | 如果上下文没有被密钥值初始化 |
| **Header file** | #include "ara/crypto/cryp/signer\_private\_ctx.h" | |
| **Description** | 签名直接提供的散列值或消息值。该方法可用于实现直接处理消息的“多重传递”签名算法，即不需要“预哈希”(如Ed25519ctx)。该方法也适用于传统的预哈希签名方案(如Ed25519ph、Ed448ph、ECDSA)的实现。如果目标算法不支持context参数，则必须提供空值(默认值)! | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00236] SignPreHashed

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00236 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23513 SWS\_CRYPT\_02415 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SignPreHashed(AlgId hashAlgId, ReadOnlyMemRegion hashValue, ReadOnlyMemRegion context=ReadOnlyMemRegion()) | |
| **Scope** | class ara::crypto::cryp::SignerPrivateCtx | |
| **Syntax** | virtual ara::core::Result<Signature::Uptrc> SignPreHashed (AlgId hash AlgId, ReadOnlyMemRegion hashValue, ReadOnlyMemRegion context=ReadOnly MemRegion()) const noexcept=0; | |
| **Parameters (in)** | hashAlgId | 哈希功能算法ID |
| hashValue | 哈希函数值(没有任何截断的结果摘要) |
| context | 用户提供的可选“上下文”(它的支持取决于具体的算法) |
| **Return value** | ara::core::Result< Signature::Uptrc > | 指向序列化签名的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 如果哈希函数算法不符合此上下文的签名算法规范 |
| CryptoErrorDomain::kInvalidInputSiz | 如果用户提供的上下文大小不正确(或不支持) |
|  | CryptoErrorDomain::kUninitialized Context | 此上下文未由密钥值初始化 |
| **Header file** | #include "ara/crypto/cryp/signer\_private\_ctx.h" | |
| **Description** | 签名一个直接提供的摘要值，并创建签名对象。此方法必须将所用密钥对的哈希函数算法ID和COUID放到结果签名对象中!用户提供的上下文可用于以下算法:Ed25519ctx, Ed25519ph, Ed448ph。如果目标算法不支持context参数，则必须提供空值(默认值)! | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00237] Sign

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00237 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23514 SWS\_CRYPT\_02416 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Sign(ReadOnlyMemRegion value, ReadOnlyMemRegion context=ReadOnlyMemRegion()) | |
| **Scope** | class ara::crypto::cryp::SignerPrivateCtx | |
| **Syntax** | template <typename Alloc = <implementation-defined>> ara::core::Result<ByteVector<Alloc> > Sign (ReadOnlyMemRegion value, ReadOnlyMemRegion context=ReadOnlyMemRegion()) const noexcept; | |
|  | Alloc | 输出容器的自定义分配器类型 |
| **Parameters (in)** | value | 应签名的(预)散列或直接消息值 |
|  | context | 用户提供的可选“上下文”(它的支持取决于具体的算法) |
| **Return value** | ara::core::Result< ByteVector< Alloc > > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidInputSize | 如果输入值或上下文参数的大小不正确/不受支持 |
| CryptoErrorDomain::kInsufficient Capacity | 如果输出签名容器的容量不足 |
|  | CryptoErrorDomain::kUninitialized Context | 如果上下文没有被密钥值初始化 |
| **Header file** | #include "ara/crypto/cryp/signer\_private\_ctx.h" | |
| **Description** | 签名直接提供的散列值或消息值。该方法可用于实现直接处理消息的“多重传递”签名算法，即不需要“预哈希”(如Ed25519ctx)。该方法也适用于传统的预哈希签名方案(如Ed25519ph、Ed448ph、ECDSA)的实现。这个方法根据实际保存的值来设置输出容器的大小!如果目标算法不支持context参数，则必须提供空值(默认值)! | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00238] CountBytesInCache

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00238 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23620 SWS\_CRYPT\_01658 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CountBytesInCache() | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | virtual std::size\_t CountBytesInCache () const noexcept=0; | |
| **Return value** | std::size\_t | 上下文缓存中保存的字节数 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 计算现在保持在上下文缓存中的字节数。在按块模式下，如果应用程序提供了最后一个不完整的输入数据块，那么上下文将最后一个(不完整的)块的其余部分保存到内部“缓存”内存中，并等待下一次调用额外的输入来完成这个块。 | |
| **Additional** |  | |

(RS\_CRYPTO\_02302)

#### [SWRD-API-Crypto-00239] EstimateMaxInputSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00239 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23621 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | EstimateMaxInputSize(std::size\_t outputCapacity) | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | std::size\_t EstimateMaxInputSize (std::size\_t outputCapacity) const noexcept; | |
| **Parameters (in)** | outputCapacity | 输出缓冲区的容量 |
| **Return value** | std::size\_t | 最大输入字节数 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 估计用于填充输出缓冲区而不溢出的最大输入字节数 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00240] EstimateRequiredCapacity

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00240 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23622 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | EstimateRequiredCapacity(std::size\_t inputSize, bool isFinal=false) | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | std::size\_t EstimateRequiredCapacity (std::size\_t inputSize, bool is Final=false) const noexcept; | |
| **Parameters (in)** | inputSize | 输入数据的大小 |
| isFinal | 指示处理最后一个数据块的标志(如果为true) |
| **Return value** | std::size\_t | 输出缓冲区所需的容量(以字节为单位) |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 估计输出缓冲区所需的最小容量，这足以保存输入数据处理的结果。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00241] FinishBytes

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00241 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23618 SWS\_CRYPT\_01657 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | FinishBytes(ReadOnlyMemRegion in) | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > Finish Bytes (ReadOnlyMemRegion in) noexcept=0; | |
| **Parameters (in)** | in | 输入数据缓存 |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 输出数据缓存 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInsufficient Capacity | 如果输出缓冲区的容量不够 |
| CryptoErrorDomain::kInOutBuffers Intersect | 如果输入和输出缓冲区相交 |
|  | CryptoErrorDomain::kProcessingNot Started | 如果数据处理不是通过调用Start()方法启动的 |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 处理消息的最后一部分(可能没有对齐到块大小边界)。如果(Is BytewiseMode() == false)那么它必须是:bs = GetBlockSize()， out.size() >= (((in.size() + bs \* ((CryptoTransform::kEncrypt == GetTransformation().Value()) ?2:1) / bs) \* bs)如果(IsBytewiseMode() == true，那么它必须是:out.size() >= in.size().输入和输出缓冲区不能相交!在按块模式处理最后一个数据块时，必须使用这种方法!此方法可用于在单个调用中处理整个消息(在任何模式中)!在输入数据缓冲区输出数据缓冲区CryptoErrorDomain::kInsufficientCapacity如果输出缓冲区的容量是不够;CryptoErrorDomain::kInOutBuffersIntersect如果输入和输出缓冲区相交;CryptoErrorDomain:: k ProcessingNotStarted如果数据处理并不是由调用Start()方法. | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00242] FinishBytes

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00242 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23619 SWS\_CRYPT\_01657 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | FinishBytes(ReadOnlyMemRegion in) | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | template <typename Alloc = <implementation-defined>> ara::core::Result<ByteVector<Alloc> > FinishBytes (ReadOnlyMemRegion in) noexcept; | |
| **Template param** | Alloc | 输出容器的自定义分配器类型 |
| **Parameters (in)** | in | 输入数据缓冲区，该缓冲区不能指向输出容器内部! |
| **Return value** | ara::core::Result< ByteVector< Alloc > > | 输出数据的托管容器 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInsufficient Capacity | 如果输出容器的容量不够 |
| CryptoErrorDomain::kInOutBuffers Intersect | 如果输入和输出缓冲区相交 |
|  | CryptoErrorDomain::kProcessingNot Started | 如果数据处理不是通过调用Start()方法启动的 |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 处理消息的最后一部分(可能没有对齐到块大小边界)。该方法根据实际保存的值设置输出容器的大小。如果(IsBytewise模式()== false)那么它必须是:bs = GetBlockSize()， out.capacity() >= (((in.size() + bs \* ((CryptoTransform::kEncrypt == gettransform . value ()) ?2: 1) - 1) / bs) \* bs)如果(IsBytewiseMode() == true，那么它必须是:out.capacity() >= in.size()使用这个方法是在块的方式处理最后的数据块!此方法可用于在单个调用中处理整个消息(在任何模式中)! | |
| **Additional** | - | |

(RS\_CRYPTO\_02302)

#### [SWRD-API-Crypto-00243] GetBlockService

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00243 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23602 ] SWS\_CRYPT\_01651 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetBlockService() | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | virtual BlockService::Uptr GetBlockService () const noexcept=0; | |
| **Return value** | BlockService::Uptr | - |
| **Exception Safety** | noexcept | |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 获得BlockService实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00244] IsBytewiseMode

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00244 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23611 SWS\_CRYPT\_01661 | |
| **CR** |  | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | IsBytewiseMode() | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | virtual bool IsBytewiseMode () const noexcept=0; | |
| **Return value** | bool | 如果模式可以逐字节处理消息(不填充到块边界)，则为True;如果只能逐块处理消息，则为false(只能处理完整块，填充是强制的) |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 检查字节属性的操作模式。 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-00245] GetTransformation

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00245 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23624 SWS\_CRYPT\_01660 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetTransformation() | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | virtual ara::core::Result<CryptoTransform> GetTransformation () const noexcept=0; | |
| **Return value** | ara::core::Result< CryptoTransform > | CryptoTransform |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUninitialized Context | 如果此上下文的转换方向在初始化期间是可配置的，但上下文尚未初始化 |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 获取为此上下文配置的转换类型:kEncrypt或kDecrypt。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00246] IsSeekableMode

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00246 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23612 SWS\_CRYPT\_01662 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | IsSeekableMode() | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | virtual bool IsSeekableMode () const noexcept=0; | |
| **Return value** | bool | True表示当前模式支持查找操作，false表示其他模式支持查找操作 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 检查当前模式下是否支持查找操作。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00247] ProcessBlocks

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00247 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23614 User\_defined\_00019 SWS\_CRYPT\_01655 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ProcessBlocks(ReadOnlyMemRegion in) | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > Process Blocks (ReadOnlyMemRegion in) noexcept=0; | |
| **Parameters (in)** | in | 输入数据缓存 |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 输出数据缓存 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncompatible Arguments | 如果输入和输出缓冲区的大小不相等 |
| CryptoErrorDomain::kInvalidInputSize | 如果输入缓冲区的大小不能被块大小整除(参见GetBlockSize()) |
|  | CryptoErrorDomain::kInOutBuffers Intersect | 如果输入和输出缓冲区部分相交 |
|  | CryptoErrorDomain::kInvalidUsage Order | 如果在处理非对齐数据(到块大小边界)后调用此方法 |
|  | CryptoErrorDomain::kProcessingNot Started | 如果数据处理不是通过调用Start()方法启动的 |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 处理与块大小边界对齐的消息的初始部分。这是一个复制优化的方法，不使用内部缓存缓冲区!它只能在处理任何非对齐块大小的边界数据之前使用。指向输入和输出缓冲区的指针必须与块大小边界对齐!输入和输出缓冲区可以完全重合，但它们不能部分相交! | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00248] ProcessBlocks

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00248 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23615 User\_defined\_00019 SWS\_CRYPT\_01655 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ProcessBlocks(ReadWriteMemRegion inOut) | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | virtual ara::core::Result<void> ProcessBlocks (ReadWriteMemRegion in Out) noexcept=0; | |
| **Parameters (in)** | inOut | 一个输入和输出数据缓冲区，即整个缓冲区应该被更新 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidInputSize | 如果inOut缓冲区的大小不能被块大小整除(参见GetBlockSize()) |
| CryptoErrorDomain::kInvalidUsage Order | 如果在处理非对齐数据(到块大小边界)后调用此方法 |
|  | CryptoErrorDomain::kProcessingNot Started | 如果数据处理不是通过调用Start()方法启动的 |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 处理与块大小边界对齐的消息的初始部分。这是一个复制优化的方法，不使用内部缓存缓冲区!它可用于第一次非块对齐的数据处理。指向输入输出缓冲区的指针必须与块大小边界对齐! | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00249] ProcessBytes

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00249 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23616 SWS\_CRYPT\_01656 | |
| **CR** |  | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SWS\_CRYPT\_23616 | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > ProcessBytes (ReadOnlyMemRegion in) noexcept=0; | |
| **Parameters (in)** | in | 输入数据缓存 |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 输出数据缓存 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInsufficient Capacity | 如果输出缓冲区的容量不足以放置转换结果 |
| CryptoErrorDomain::kInOutBuffers Intersect | 如果输入和输出缓冲区相交 |
|  | CryptoErrorDomain::kProcessingNot Started | 如果数据处理不是通过调用Start()方法启动的 |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 处理消息的非最后部分(未与块大小边界对齐)。如果(IsBytewiseMode() == false)那么它必须是:bs = GetBlockSize()， out.size() >= (((in.size() + bs 1) / bs) \* bs)如果(IsBytewiseMode() == true)那么它必须是:out.size() >= in.size()输入和输出缓冲区不能相交!这个方法是“复制无效”的，因此它应该只在应用程序不能控制原始消息的分块的情况下使用! | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00250] ProcessBytes

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00250 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23617 SWS\_CRYPT\_01656 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ProcessBytes(ReadOnlyMemRegion in) | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | template <typename Alloc = <implementation-defined>> ara::core::Result<ByteVector<Alloc> > ProcessBytes (ReadOnlyMemRegion in) noexcept; | |
| **Template param** | Alloc | 输出容器的自定义分配器类型 |
| **Parameters (in)** | in | 输入数据缓存 |
| **Return value** | ara::core::Result< ByteVector< Alloc > > | 输出数据的托管容器 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInsufficient Capacity | 如果输出容器的容量不够 |
| CryptoErrorDomain::kInOutBuffers Intersect | 如果输入缓冲指向在预先分配的输出容器内 |
|  | CryptoErrorDomain::kProcessingNotStarted | 如果数据处理不是通过调用Start()方法启动的 |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 处理消息的非最终部分(未与块大小边界对齐)。该方法根据实际保存的值设置输出容器的大小。如果(IsBytewiseMode() == false)则必须是:bs = GetBlockSize()， out.capacity() >= ((in.size() + bs - 1) / bs) \* bs)如果(IsBytewiseMode() == true，则必须是:out.capacity() >= in.size()这个方法是“copy inefficient”的，因此它应该只在应用程序不能控制原始消息分块的情况下使用!输入缓冲区不能指向输出容器内部! | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00251] Reset

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00251 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23627 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Reset() | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | virtual ara::core::Result<void> Reset () noexcept=0; | |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 清除上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00252] Seek

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00252 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23613 SWS\_CRYPT\_01653 | |
| **CR** | -- | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Seek(std::int64\_t offset, bool fromBegin=true) | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | virtual ara::core::Result<void> Seek (std::int64\_t offset, bool from Begin=true) noexcept=0; | |
| **Parameters (in)** | offset | 以字节为单位的偏移值，相对于伽马流的开始位置或当前位置 |
| fromBegin | 在流中定位的起点:从begin开始(如果为true)或从当前位置开始(如果为false) |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUnsupported | 如果当前模式不支持查找操作 |
| CryptoErrorDomain::kProcessingNot Started | 如果数据处理不是通过调用Start()方法启动的 |
|  | CryptoErrorDomain::kBelowBoundary | 如果偏移值不正确(在fromBegin参数的上下文中)，即它指向流的开始之前(注意:这是一个可选的错误条件) |
|  | CryptoErrorDomain::kInvalidArgument | 如果偏移量没有对齐到所需的边界(参见IsBytewiseMode()) |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 设置加密/解密伽马的流中的下一个字节的位置。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00253] SetKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00253 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23623 SWS\_CRYPT\_01659 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SetKey(const SymmetricKey &key, CryptoTransform transform=CryptoTransform::kEncrypt) | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | virtual ara::core::Result<void> SetKey (const SymmetricKey &key, CryptoTransform transform=CryptoTransform::kEncrypt) noexcept=0; | |
| **Parameters (in)** | key | 源密钥对象 |
| transform | 方向”指示器:部署用于直接转换(如果为true)或反向转换(如果为false)的密钥 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncompatible Object | 如果提供的键对象与这个对称键上下文不兼容 |
| CryptoErrorDomain::kUsageViolation | 如果与此上下文关联的转换类型(考虑到转换指定的方向)被所提供的密钥对象的“允许使用”限制所禁止 |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 设置(部署)一个密钥到流加密算法上下文。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00254] Start

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00254 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23625 SWS\_CRYPT\_01654 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Start(ReadOnlyMemRegion iv=ReadOnlyMemRegion()) | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | virtual ara::core::Result<void> Start (ReadOnlyMemRegion iv=ReadOnly MemRegion()) noexcept=0; | |
| **Parameters (in)** | iv | 一个可选的初始化向量(IV)或“nonce”值 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUninitialized Context | 如果没有通过部署密钥初始化上下文 |
| CryptoErrorDomain::kInvalidInputSize | 如果所提供的IV的大小不受支持(即，如果它对初始化来说不够) |
|  | CryptoErrorDomain::kUnsupported | 如果基本算法(或其当前实现)主要不支持IV变量，但IV值不为空，即if (IV .empty() == false) |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 为新的数据流处理或生成(取决于原语)初始化上下文。如果IV大小大于算法最大支持的大小，那么实现可以只使用序列的前导字节。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00255] Start

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00255 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23626 SWS\_CRYPT\_01654 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Start(const SecretSeed &iv) | |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx | |
| **Syntax** | virtual ara::core::Result<void> Start (const SecretSeed &iv) noexcept=0; | |
| **Parameters (in)** | iv |  |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUninitialized Context | 如果没有通过部署密钥初始化上下文 |
| CryptoErrorDomain::kInvalidInputSize | 如果所提供的IV的大小不受支持(即，如果它对初始化来说不够) |
|  | CryptoErrorDomain::kUnsupported | 如果基本算法(或其当前实现)主要不支持IV变量 |
|  | CryptoErrorDomain::kUsageViolation | 如果该转换类型被提供的Secret Seed对象的“允许使用”限制所禁止 |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" | |
| **Description** | 为新的数据流处理或生成(取决于原语)初始化上下文。如果IV大小大于算法最大支持的大小，那么实现可以只使用序列的前导字节。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00256] GetCryptoService

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00256 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23702 SWS\_CRYPT\_01506 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetCryptoService() | |
| **Scope** | class ara::crypto::cryp::SymmetricBlockCipherCtx | |
| **Syntax** | virtual CryptoService::Uptr GetCryptoService () const noexcept=0; | |
| **Return value** | CryptoService::Uptr |  |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/symmetric\_block\_cipher\_ctx.h" | |
| **Description** | 获得CryptoService实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00257 GetTransformation

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00257 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23711 SWS\_CRYPT\_01508 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetTransformation() | |
| **Scope** | class ara::crypto::cryp::SymmetricBlockCipherCtx | |
| **Syntax** | virtual ara::core::Result<CryptoTransform> GetTransformation () const noexcept=0; | |
| **Return value** | ara::core::Result< CryptoTransform > | CryptoTransform |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUninitialized Context, | 如果SetKey()还没有被调用。 |
| **Header file** | #include "ara/crypto/cryp/symmetric\_block\_cipher\_ctx.h" | |
| **Description** | 获取为此上下文配置的转换类型:kEncrypt或kDecrypt。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00258] ProcessBlock

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00258 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23716 SWS\_CRYPT\_01503 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ProcessBlock(ReadOnlyMemRegion in, bool suppressPadding=false) | |
| **Scope** | class ara::crypto::cryp::SymmetricBlockCipherCtx | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > Process Block (ReadOnlyMemRegion in, bool suppressPadding=false) const noexcept=0; | |
| **Parameters (in)** | in | 输入数据块 |
| suppressPadding | 如果为true，则此方法不应用填充，因此输入缓冲区的大小与块大小相同，即，要么待处理的数据完全符合块大小，要么用户必须应用填充达到相同的效果。 |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 包含转换结果的输出缓冲区 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidInputSize | 如果布尔类型参数 {suppressPadding}被设置为TRUE，并且提供的输入缓冲区不匹配块大小。 |
| CryptoErrorDomain::kUninitialized Context | 如果上下文没有通过调用SetKey()初始化 |
| **Header file** | #include "ara/crypto/cryp/symmetric\_block\_cipher\_ctx.h" | |
| **Description** | 根据配置处理(加密/解密)一个输入块 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00259] ProcessBlock

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00259 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23717 SWS\_CRYPT\_01503 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ProcessBlock(ReadOnlyMemRegion in, bool suppressPadding=false) | |
| **Scope** | class ara::crypto::cryp::SymmetricBlockCipherCtx | |
| **Syntax** | template <typename Alloc = <implementation-defined>> ara::core::Result<ByteVector<Alloc> > ProcessBlock (ReadOnlyMemRegion in, bool suppressPadding=false) const noexcept; | |
| **Template param** | Alloc | 输出容器的自定义分配器类型 |
| **Parameters (in)** | in | 输入数据块 |
| suppressPadding | 如果为true，则该方法不应用填充，但有效载荷应该填充普通数据的整个块 |
| **Return value** | ara::core::Result< ByteVector< Alloc > > | 输出块的托管容器 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncorrectInput Size | 如果违反了上面提到的关于输入大小的规则 |
| CryptoErrorDomain::kInsufficient Capacity | 如果out.size()不足以存储转换结果 |
|  | CryptoErrorDomain::kUninitialized Context | 如果上下文没有被密钥值初始化 |
| **Header file** | #include "ara/crypto/cryp/symmetric\_block\_cipher\_ctx.h" | |
| **Description** | 根据配置处理(加密/解密)一个输入块。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00260] ProcessBlocks

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00260 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23715 SWS\_CRYPT\_01504 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ProcessBlocks(ReadOnlyMemRegion in) | |
| **Scope** | class ara::crypto::cryp::SymmetricBlockCipherCtx | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > ProcessBlocks (ReadOnlyMemRegion in) const noexcept=0; | |
| **Parameters (in)** | in | 输入数据缓存 |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 输出数据缓存 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUninitialized Context | 如果上下文没有被密钥初始化 |
| CryptoErrorDomain::kInvalidInputSize | 如果输入缓冲区的大小不能被块大小整除(参见GetBlockSize()) |
| **Header file** | #include "ara/crypto/cryp/symmetric\_block\_cipher\_ctx.h" | |
| **Description** | 根据配置处理(加密/解密)一个输入块。in的大小必须能被块大小整除(参见GetBlockSize())。指向输入缓冲区的指针必须与块大小的边界对齐! | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00261] Reset

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00261 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23712 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Reset() | |
| **Scope** | class ara::crypto::cryp::SymmetricBlockCipherCtx | |
| **Syntax** | virtual ara::core::Result<void> Reset () noexcept=0; | |
| **Return value** | ara::core::Result< void > | 如果转换需要输入数据的最大大小，则为True，否则为false |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUninitialized Context | 如果此上下文的转换方向在初始化期间是可配置的，但上下文尚未初始化 |
| **Header file** | #include "ara/crypto/cryp/symmetric\_block\_cipher\_ctx.h" | |
| **Description** | 指示当前配置的转换只接受完整的输入数据块。清除加密上下文。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00262] SetKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00262 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_23710 SWS\_CRYPT\_01501 SWS\_CRYPT\_01502 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SetKey(const SymmetricKey &key, CryptoTransform transform=CryptoTransform::kEncrypt) | |
| **Scope** | class ara::crypto::cryp::SymmetricBlockCipherCtx | |
| **Syntax** | virtual ara::core::Result<void> SetKey (const SymmetricKey &key, CryptoTransform transform=CryptoTransform::kEncrypt) noexcept=0; | |
| **Parameters (in)** | key | 源密钥对象 |
| transform | 方向”指示器:部署用于直接转换(如果为true)或反向转换(如果为false)的密钥 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncompatible Object | 如果提供的密钥对象属于不同的CryptoProvider实例 |
| CryptoErrorDomain::kUsageViolation | 如果与此上下文关联的转换类型(考虑到转换指定的方向)被所提供的密钥对象的“允许使用”限制所禁止 |
| **Header file** | #include "ara/crypto/cryp/symmetric\_block\_cipher\_ctx.h" | |
| **Description** | 为对称算法上下文设置(部署)一个密钥 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00263] CalculateWrappedKeySize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00263 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24013 SWS\_CRYPT\_02121 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | CalculateWrappedKeySize(std::size\_t keyLength) | |
| **Scope** | class ara::crypto::cryp::SymmetricKeyWrapperCtx | |
| **Syntax** | virtual std::size\_t CalculateWrappedKeySize (std::size\_t keyLength) const noexcept=0; | |
| **Parameters (in)** | keyLength | 以比特为单位的原始密钥长度 |
| **Return value** | std::size\_t | 包装密钥的大小(以字节为单位) |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/symmetric\_key\_wrapper\_ctx.h" | |
| **Description** | 从原始密钥长度(位)计算封装密钥的大小(字节)。这个方法对于一些不同于RFC3394 / RFC5649的实现是有用的。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00264] GetExtensionService

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00264 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24002 | |
| **CR** |  | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetExtensionService() | |
| **Scope** | class ara::crypto::cryp::SymmetricKeyWrapperCtx | |
| **Syntax** | virtual ExtensionService::Uptr GetExtensionService () const noexcept=0; | |
| **Return value** | ExtensionService::Uptr | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/symmetric\_key\_wrapper\_ctx.h" | |
| **Description** | 获得ExtensionService实例 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00266] GetMaxTargetKeyLength

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00265 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24012 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetMaxTargetKeyLength() | |
| **Scope** | class ara::crypto::cryp::SymmetricKeyWrapperCtx | |
| **Syntax** | virtual std::size\_t GetMaxTargetKeyLength () const noexcept=0; | |
| **Return value** | std::size\_t | 目标密钥的最大长度(以比特为单位) |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/symmetric\_key\_wrapper\_ctx.h" | |
| **Description** | 取实现支持的目标密钥的最大长度。这个方法对于一些不同于RFC3394 / RFC5649的实现是有用的。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00266] GetTargetKeyGranularity

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00266 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24011 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetTargetKeyGranularity() | |
| **Scope** | class ara::crypto::cryp::SymmetricKeyWrapperCtx | |
| **Syntax** | virtual std::size\_t GetTargetKeyGranularity () const noexcept=0; | |
| **Return value** | std::size\_t | 块的大小(以字节表示) |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/symmetric\_key\_wrapper\_ctx.h" | |
| **Description** | 获得目标密钥的预期粒度(块大小)。如果类实现了RFC3394 (KW没有填充)，那么这个方法应该返回8(即8个字节= 64位)。如果类实现了RFC5649(带有填充的KW)，那么这个方法应该返回1(即1八位= 8位) | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00267] Reset

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00267 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24019 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Reset() | |
| **Scope** | class ara::crypto::cryp::SymmetricKeyWrapperCtx | |
| **Syntax** | virtual ara::core::Result<void> Reset () noexcept=0; | |
| **Return value** | ara::core::Result< void > |  |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/symmetric\_key\_wrapper\_ctx.h" | |
| **Description** | 清除加密上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00268] SetKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00268 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24018 SWS\_CRYPT\_02122 SWS\_CRYPT\_02123 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SetKey(const SymmetricKey &key, CryptoTransform transform) | |
| **Scope** | class ara::crypto::cryp::SymmetricKeyWrapperCtx | |
| **Syntax** | virtual ara::core::Result<void> SetKey (const SymmetricKey &key, CryptoTransform transform) noexcept=0; | |
| **Parameters (in)** | key | 源密钥对象 |
| transform | “方向”指示器:部署用于直接转换(如果为true)或反向转换(如果为false)的密钥 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kIncompatible Object | 如果提供的密钥对象与这个对称密钥上下文不兼容 |
| CryptoErrorDomain::kUsageViolation | 如果与此上下文关联的转换类型(考虑到转换指定的方向)被所提供的密钥对象的“允许使用”限制所禁止 |
| **Header file** | #include "ara/crypto/cryp/symmetric\_key\_wrapper\_ctx.h" | |
| **Description** | 将一个密钥设置(部署)到对称密钥包装器算法上下文。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00269] UnwrapConcreteKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00269 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24017 SWS\_CRYPT\_02109 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | UnwrapConcreteKey(ReadOnlyMemRegion wrappedKey, AlgId algId, AllowedUsageFlags allowedUsage) | |
| **Scope** | class ara::crypto::cryp::SymmetricKeyWrapperCtx | |
| **Syntax** | template <typename ExpectedKey> ara::core::Result<typename ExpectedKey::Uptrc> UnwrapConcreteKey (Read OnlyMemRegion wrappedKey, AlgId algId, AllowedUsageFlags allowedUsage) noexcept | |
| **Template param** | ExpectedKey | 密钥的期望类型 |
| **Parameters (in)** | wrappedKey | 包含待包装密钥内存区 |
| algId | 目标对称加密算法标识符 |
| allowedUsage | 位标志，定义允许的转换类型列表，在其中可以使用目标密钥 |
| **Return value** | ara::core::Result< typename Expected Key::Uptrc | 指向ExpectedKey对象的唯一智能指针，它保持解包的密钥材料 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidInputSize | 如果所提供的包装密钥的大小不受支持 |
| CryptoErrorDomain::kUninitialized Context | 如果上下文没有使用密钥值初始化 |
| **Header file** | #include "ara/crypto/cryp/symmetric\_key\_wrapper\_ctx.h" | |
| **Description** | 为所提供的BLOB执行“key unwrap”操作，并生成预期类型的key对象。更多细节请参见UnwrapKey() | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00270] UnwrapKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00270 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24016 SWS\_CRYPT\_02107 SWS\_CRYPT\_02109 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | UnwrapKey(ReadOnlyMemRegion wrappedKey, AlgId algId, AllowedUsageFlags allowed Usage) | |
| **Scope** | class ara::crypto::cryp::SymmetricKeyWrapperCtx | |
| **Syntax** | virtual ara::core::Result<RestrictedUseObject::Uptrc> UnwrapKey (Read OnlyMemRegion wrappedKey, AlgId algId, AllowedUsageFlags allowedUsage) const noexcept=0; | |
| **Parameters (in)** | wrappedKey | 包含待包装密钥内存区 |
| algId | 目标对称加密算法标识符 |
| allowedUsage | 位标志，定义允许的转换类型列表，在其中可以使用目标密钥 |
| **Return value** | ara::core::Result< RestrictedUse Object::Uptrc > | Key对象的唯一智能指针，它保持解包的密钥材料 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidInputSize | 如果所提供的包装密钥的大小不受支持 |
| CryptoErrorDomain::kUninitialized Context | 如果上下文没有使用密钥值初始化 |
| **Header file** | #include "ara/crypto/cryp/symmetric\_key\_wrapper\_ctx.h" | |
| **Description** | 对提供的BLOB执行“key unwrap”操作并生成key对象。如果实现基于AES分组密码并应用于AES密钥，则该方法应符合RFC3394或RFC5649。创建的Key对象有以下属性:临时和不可导出(因为它是在没有元信息的情况下导入的)!SymmetricKey::Uptrc key = SymmetricKey::Cast(Unwrap key (wrappedKey，…));PrivateKey可以通过以下方式来解包: PrivateKey::Uptrc key = PrivateKey::Cast(UnwrapKey(wrappedKey, ...));在这两个例子中，如果打开的密钥的实际类型与目标密钥的类型不同，则Cast()方法可能会抛出BadObjectTypeException ! | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00271] UnwrapSeed

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00271 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24015 SWS\_CRYPT\_02108 SWS\_CRYPT\_02109 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | UnwrapSeed(ReadOnlyMemRegion wrappedSeed, AlgId targetAlgId, SecretSeed::Usage allowedUsage) | |
| **Scope** | class ara::crypto::cryp::SymmetricKeyWrapperCtx | |
| **Syntax** | virtual ara::core::Result<SecretSeed::Uptrc> UnwrapSeed (ReadOnlyMem Region wrappedSeed, AlgId targetAlgId, SecretSeed::Usage allowedUsage) const noexcept=0; | |
| **Parameters (in)** | wrappedSeed | 包含被包装的种子的内存区域 |
| targetAlgId | 目标对称算法标识符(也定义了目标种子长度) |
| allowedUsage | 允许的目标种子的使用范围 |
| **Return value** | ara::core::Result< SecretSeed::Uptrc > | SecretSeed对象的唯一智能指针，它保持解包的密钥材料 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidInputSize | 如果所提供的包装种子的大小不受支持 |
| CryptoErrorDomain::kUninitialized Context | 如果上下文没有使用密钥值初始化 |
| **Header file** | #include "ara/crypto/cryp/symmetric\_key\_wrapper\_ctx.h" | |
| **Description** | 执行提供的BLOB的“key unwrap”操作并生成SecretSeed对象。如果实现基于AES分组密码并应用于AES密钥材料，则该方法应符合RFC3394或RFC5649。创建的SecretSeed对象具有以下属性:临时和不可导出(因为它是在没有元信息的情况下导入的)。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00272] WrapKeyMaterial

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00272 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24014 SWS\_CRYPT\_02105 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | WrapKeyMaterial(const RestrictedUseObject &key) | |
| **Scope** | class ara::crypto::cryp::SymmetricKeyWrapperCtx | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > WrapKey Material (const RestrictedUseObject &key) const noexcept=0; | |
| **Parameters (in)** | key | 待封装的密钥 |
| **Return value** | ara::core::Result< ara::core::Vector< ara::core::Byte > > |  |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInsufficient Capacity | 如果包装的缓冲区的大小不足以存储结果 |
| CryptoErrorDomain::kInvalidInputSize | 如果密钥对象的长度不受支持 |
|  | CryptoErrorDomain::kUninitialized Context | 如果上下文没有被密钥值初始化 |
| **Header file** | #include "ara/crypto/cryp/symmetric\_key\_wrapper\_ctx.h" | |
| **Description** | 对所提供的关键材料执行“密钥包装”操作。如果一个实现基于AES分组密码并应用于AES密钥，那么这个方法应该符合RFC3394或RFC5649。方法CalculateWrappedKeySize()可用于计算所需输出缓冲区的大小。 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-00273] GetSignatureService

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00273 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24102 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetSignatureService() | |
| **Scope** | class ara::crypto::cryp::VerifierPublicCtx | |
| **Syntax** | virtual SignatureService::Uptr GetSignatureService () const noexcept=0; | |
| **Return value** | SignatureService::Uptr | - |
| **Exception Safety** | noexcept | |
| **Header file** | #include "ara/crypto/cryp/verifier\_public\_ctx.h" | |
| **Description** | 获取SignatureService实例 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-00274] Reset

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00274 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24116 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Reset() | |
| **Scope** | class ara::crypto::cryp::VerifierPublicCtx | |
| **Syntax** | virtual ara::core::Result<void> Reset () noexcept=0; | |
| **Return value** | ara::core::Result<void> | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/cryp/verifier\_public\_ctx.h" | |
| **Description** | 清除加密上下文 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00275] SetKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00275 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24115 SWS\_CRYPT\_01821 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SetKey(const PublicKey &key) | |
| **Scope** | class ara::crypto::cryp::VerifierPublicCtx | |
| **Syntax** | virtual ara::core::Result<void> SetKey (const PublicKey &key) noexcept=0; | |
| **Parameters (in)** | key |  |
| **Return value** | ara::core::Result< void > |  |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrc::kIncompatibleObject | 如果提供的密钥对象与这个上下文不兼容 |
| CryptoErrc::kUsageViolation | 如果与此上下文关联的转换类型被所提供的密钥对象的“允许使用”限制所禁止 |
| **Header file** | #include "ara/crypto/cryp/verifier\_public\_ctx.h" | |
| **Description** | 置(部署)一个密钥到验证公共算法上下文。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00276] VerifyPrehashed

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00276 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24111 SWS\_CRYPT\_02417 SWS\_CRYPT\_02418 | |
| **CR** |  | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | VerifyPrehashed(CryptoAlgId hashAlgId, ReadOnlyMemRegion hashValue, const Signature &signature, ReadOnlyMemRegion context=ReadOnlyMemRegion()) | |
| **Scope** | class ara::crypto::cryp::VerifierPublicCtx | |
| **Syntax** | virtual ara::core::Result<bool> VerifyPrehashed (CryptoAlgId hashAlg Id, ReadOnlyMemRegion hashValue, const Signature &signature, ReadOnly MemRegion context=ReadOnlyMemRegion()) const noexcept=0; | |
| **Parameters (in)** | hashAlgId | 哈希功能算法ID |
| hashValue | 希函数值(没有任何截断的结果摘要) |
| signature | 需要验证的签名对象 |
| context | 用户提供的可选“上下文”(它的支持取决于具体的算法) |
| **Return value** | ara::core::Result< bool > | 如果签名验证成功，则为True，否则为false |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrc::kProcessingNotFinished | 如果方法hashFn.Finish()在此方法调用之前没有被调用 |
| CryptoErrc::kInvalidArgument | 如果hashFn的CryptoAlgId与此上下文的CryptoAlgId不同 |
|  | CryptoErrc::kInvalidInputSize | 如果提供的上下文的大小与配置的签名算法不兼容。 |
| **Header file** | #include "ara/crypto/cryp/verifier\_public\_ctx.h" | |
| **Description** | 通过存储在哈希函数上下文中的摘要值验证签名。这是一个直通到SWS\_CRYPT\_24113的接口，方便开发人员使用，也就是说，它增加了额外的输入检查，然后从SWS\_CRYPT\_24113调用verify()接口。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00277] Verify

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00277 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24112 SWS\_CRYPT\_02419 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Verify(ReadOnlyMemRegion value, ReadOnlyMemRegion signature, ReadOnlyMemRegion context=ReadOnlyMemRegion()) | |
| **Scope** | class ara::crypto::cryp::VerifierPublicCtx | |
| **Syntax** | virtual ara::core::Result<bool> Verify (ReadOnlyMemRegion value, ReadOnlyMemRegion signature, ReadOnlyMemRegion context=ReadOnlyMem Region()) const noexcept=0; | |
| **Parameters (in)** | value | 应进行验证的(预)散列或直接消息值 |
| signature | 用于验证的签名BLOB(该BLOB包含位于算法规范定义的固定/最大长度字段中的数字签名组件的普通序列，每个组件由一个原始字节序列表示，该序列由0填充到字段的全长度;例如，在(EC)DSA-256的情况下(即q模块的长度为256位)，签名BLOB必须有两个固定大小的字段:32 + 32字节，R和S组件分别为32 + 32字节，即BLOB总大小为64字节) |
| context | 用户提供的可选“上下文”(它的支持取决于具体的算法) |
| **Return value** | ara::core::Result< bool > | 如果签名验证成功，则为True，否则为false |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrc::kUninitializedContext | 上下文没有通过密钥初始化 |
| CryptoErrc::kInvalidInputSize | 如果context参数的大小不受支持 |
| **Header file** | #include "ara/crypto/cryp/verifier\_public\_ctx.h" | |
| **Description** | 通过直接提供的散列值或消息值验证签名BLOB。该方法可用于实现直接处理消息的“多重传递”签名算法，即不需要“预哈希”(如Ed25519ctx)。该方法也适用于传统的预哈希签名方案(如Ed25519ph、Ed448ph、ECDSA)的实现。如果目标算法不支持context参数，则必须提供空值(默认值)!用户提供的上下文可用于以下算法:Ed25519ctx, Ed25519ph, Ed448ph | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00278] VerifyPrehashed

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00278 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24113 SWS\_CRYPT\_02417 SWS\_CRYPT\_02418 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | VerifyPrehashed(const HashFunctionCtx &hashFn, const Signature &signature, ReadOnlyMem Region context=ReadOnlyMemRegion()) | |
| **Scope** | class ara::crypto::cryp::VerifierPublicCtx | |
| **Syntax** | virtual ara::core::Result<bool> VerifyPrehashed (const HashFunctionCtx &hashFn, const Signature &signature, ReadOnlyMemRegion context=Read OnlyMemRegion()) const noexcept=0; | |
| **Parameters (in)** | hashFn | 哈希运算上下文 |
| signature | 待验证的签名对象 |
| Context | 用户提供的可选“上下文”(它的支持取决于具体的算法) |
| **Return value** | ara::core::Result< bool > | 如果签名验证成功，则为True，否则为false |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrc::kIncompatibleObject | 如果该上下文的CryptoAlgId与签名的CryptoAlgId不匹配;或者该哈希所需的CryptoAlgId不是kAlgIdDefault，并且该上下文所需的哈希CryptoAlgId不匹配hashAlgId或签名的哈希CryptoAlgId |
| CryptoErrc::kIncompatibleArguments | 如果提供的hashAlgId不是kAlgIdDefault，并且提供的签名对象的CryptoAlgId与提供的hashAlgId不匹配 |
|  | CryptoErrc::kBadObjectReference | 如果提供的签名对象没有引用加载到上下文中的公钥，也就是说，如果上下文中公钥的COUID不等于从签名对象引用的COUID。 |
|  | CryptoErrc::kInvalidInputSize | 如果提供的上下文或hashValue的大小与配置的签名算法不兼容。 |
| **Header file** | #include "ara/crypto/cryp/verifier\_public\_ctx.h" | |
| **Description** | 通过存储在哈希函数上下文中的摘要值验证签名。这是SWS\_CRYPT\_24112的直通接口，方便开发人员使用，也就是说，它增加了额外的输入检查，然后调用默认的verify()接口。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00279] VerifyPrehashed

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00279 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_24114 SWS\_CRYPT\_02417 SWS\_CRYPT\_02418 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | VerifyPrehashed(const HashFunctionCtx &hashFn, ReadOnlyMemRegion signature, ReadOnly MemRegion context=ReadOnlyMemRegion()) | |
| **Scope** | class ara::crypto::cryp::VerifierPublicCtx | |
| **Syntax** | virtual ara::core::Result<bool> VerifyPrehashed (const HashFunctionCtx &hashFn, ReadOnlyMemRegion signature, ReadOnlyMemRegion context=Read OnlyMemRegion()) const noexcept=0; | |
| **Parameters (in)** | hashFn | 哈希计算上下文 |
| signature | 待验证的签名数据块 |
| context | 用户提供的可选“上下文”(它的支持取决于具体的算法) |
| **Return value** | ara::core::Result< bool > | 如果签名验证成功，则为True，否则为false |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrc::kProcessingNotFinished | 如果方法hashFn.Finish()在此方法调用之前没有被调用 |
| CryptoErrc::kInvalidArgument | 如果hashFn的CryptoAlgId与此上下文的CryptoAlgId不同 |
|  | CryptoErrc::kInvalidInputSize | 如果提供的上下文或签名的大小与配置的签名算法不兼容。 |
| **Header file** | #include "ara/crypto/cryp/verifier\_public\_ctx.h" | |
| **Description** | 通过存储在哈希函数上下文中的摘要值验证签名。这是SWS\_CRYPT\_24112的直通接口，方便开发人员使用，也就是说，它增加了额外的输入检查，然后调用默认的verify()接口 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-00280] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00280 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_24101 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::VerifierPublicCtx |
| **Derived from** | std::unique\_ptr<VerifierPublicCtx> |
| **Syntax** | using Uptr = std::unique\_ptr<VerifierPublicCtx>; |
| **Header file** | #include "ara/crypto/cryp/verifier\_public\_ctx.h" |
| **Description** | 接口唯一智能指针。 |
| **Additional** | - |

#### [SWRD-API-Crypto-00281] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00281 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20101 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::AuthCipherCtx |
| **Derived from** | std::unique\_ptr<AuthCipherCtx> |
| **Syntax** | using Uptr = std::unique\_ptr<AuthCipherCtx>; |
| **Header file** | #include "ara/crypto/cryp/auth\_cipher\_ctx.h" |
| **Description** | 接口唯一智能指针。 |
| **Additional** | - |

#### [SWRD-API-Crypto-00282] Uptrc

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00282 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_24802 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptrc |
| **Scope** | class ara::crypto::cryp::RestrictedUseObject |
| **Derived from** | std::unique\_ptr<const RestrictedUseObject> |
| **Syntax** | using Uptrc = std::unique\_ptr<const RestrictedUseObject>; |
| **Header file** | #include "ara/crypto/cryp/cryobj/restricted\_use\_object.h" |
| **Description** | 接口唯一智能指针。 |
| **Additional** | - |

#### [SWRD-API-Crypto-00283] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00283 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_29031 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::BlockService |
| **Derived from** | std::unique\_ptr<BlockService> |
| **Syntax** | using Uptr = std::unique\_ptr<BlockService>; |
| **Header file** | #include "ara/crypto/cryp/block\_service.h" |
| **Description** | 接口唯一智能指针。 |
| **Additional** | - |

#### [SWRD-API-Crypto-00284] AlgId

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00284 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20402 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | AlgId |
| **Scope** | class ara::crypto::cryp::CryptoContext |
| **Derived from** | CryptoAlgId |
| **Syntax** | using AlgId = CryptoAlgId; |
| **Header file** | #include "ara/crypto/cryp/crypto\_context.h" |
| **Description** | 厂商特定二进制加密原始ID的类型定义。 |
| **Additional** | - |

#### [SWRD-API-Crypto-00285] COIdentifier

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00285 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20504 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | struct |
| **Symbol** | COIdentifier |
| **Scope** | class ara::crypto::cryp::CryptoObject |
| **Syntax** | struct COIdentifier {...}; |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_object.h" |
| **Description** | 加密对象唯一标识符 |
| **Additional** | - |

#### [SWRD-API-Crypto-00286] Uptrc

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00286 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20502 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptrc |
| **Scope** | class ara::crypto::cryp::CryptoObject |
| **Derived from** | std::unique\_ptr<const CryptoObject> |
| **Syntax** | using Uptrc = std::unique\_ptr<const CryptoObject>; |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_object.h" |
| **Description** | 接口唯一智能指针。 |
| **Additional** | - |

#### [SWRD-API-Crypto-00287] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00287 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20501 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::CryptoObject |
| **Derived from** | std::unique\_ptr<CryptoObject> |
| **Syntax** | using Uptr = std::unique\_ptr<CryptoObject>; |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_object.h" |
| **Description** | 接口唯一智能指针。 |
| **Additional** | - |

#### [SWRD-API-Crypto-00288] AlgId

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00288 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20641 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | AlgId |
| **Scope** | class ara::crypto::cryp::CryptoPrimitiveId |
| **Derived from** | CryptoAlgId |
| **Syntax** | using AlgId = CryptoAlgId; |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_primitive\_id.h" |
| **Description** | 厂商特定二进制加密原始ID的类型定义。 |
| **Additional** | - |

#### [SWRD-API-Crypto-00289] Uptrc

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00289 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20644 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptrc |
| **Scope** | class ara::crypto::cryp::CryptoPrimitiveId |
| **Derived from** | std::unique\_ptr<const CryptoPrimitiveId> |
| **Syntax** | using Uptrc = std::unique\_ptr<const CryptoPrimitiveId>; |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_primitive\_id.h" |
| **Description** | 类型定义指针 |
| **Additional** |  |

#### [SWRD-API-Crypto-00290] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00290 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20643 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::CryptoPrimitiveId |
| **Derived from** | std::unique\_ptr<CryptoPrimitiveId> |
| **Syntax** | using Uptr = std::unique\_ptr<CryptoPrimitiveId>; |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_primitive\_id.h" |
| **Description** | 指向const的类型定义指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00291] AlgId

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00291 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20703 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | AlgId |
| **Scope** | class ara::crypto::cryp::CryptoProvider |
| **Derived from** | CryptoPrimitiveId::AlgId |
| **Syntax** | using AlgId = CryptoPrimitiveId::AlgId; |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" |
| **Description** | 算法ID类型定义的短别名。 |
| **Additional** | - |

#### [SWRD-API-Crypto-00292] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00292 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20701 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::CryptoProvider |
| **Derived from** | std::unique\_ptr<CryptoProvider> |
| **Syntax** | using Uptr = std::unique\_ptr<CryptoProvider>; |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" |
| **Description** | 接口共享智能指针。 |
| **Additional** | - |

#### [SWRD-API-Crypto-00293] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00293 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_29024 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::CryptoService |
| **Derived from** | std::unique\_ptr<CryptoService> |
| **Syntax** | using Uptr = std::unique\_ptr<CryptoService>; |
| **Header file** | #include "ara/crypto/cryp/crypto\_service.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00294] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00294 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20801 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::DecryptorPrivateCtx |
| **Derived from** | std::unique\_ptr<DecryptorPrivateCtx> |
| **Syntax** | using Uptr = std::unique\_ptr<DecryptorPrivateCtx>; |
| **Header file** | #include "ara/crypto/cryp/decryptor\_private\_ctx.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00295] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-ID (SWRD-ID 编号规则参见附录A-信息定义需求ID。) |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_29011 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::DigestService |
| **Derived from** | std::unique\_ptr<DigestService> |
| **Syntax** | using Uptr = std::unique\_ptr<DigestService>; |
| **Header file** | #include "ara/crypto/cryp/digest\_service.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00296] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00296 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_21001 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::EncryptorPublicCtx |
| **Derived from** | std::unique\_ptr<EncryptorPublicCtx> |
| **Syntax** | using Uptr = std::unique\_ptr<EncryptorPublicCtx>; |
| **Header file** | #include "ara/crypto/cryp/encryptor\_public\_ctx.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00297] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00297 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_29042 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::ExtensionService |
| **Derived from** | std::unique\_ptr<ExtensionService> |
| **Syntax** | using Uptr = std::unique\_ptr<ExtensionService>; |
| **Header file** | #include "ara/crypto/cryp/extension\_service.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00298] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00298 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_21101 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::HashFunctionCtx |
| **Derived from** | std::unique\_ptr<HashFunctionCtx> |
| **Syntax** | using Uptr = std::unique\_ptr<HashFunctionCtx>; |
| **Header file** | #include "ara/crypto/cryp/hash\_function\_ctx.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00299] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00299 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_21301 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::KeyAgreementPrivateCtx |
| **Derived from** | std::unique\_ptr<KeyAgreementPrivateCtx> |
| **Syntax** | using Uptr = std::unique\_ptr<KeyAgreementPrivateCtx>; |
| **Header file** | #include "ara/crypto/cryp/key\_agreement\_private\_ctx.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** |  |

#### [SWRD-API-Crypto-00300] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-ID (SWRD-ID 编号规则参见附录A-信息定义需求ID。) |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_21401 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::KeyDecapsulatorPrivateCtx |
| **Derived from** | std::unique\_ptr<KeyDecapsulatorPrivateCtx> |
| **Syntax** | using Uptr = std::unique\_ptr<KeyDecapsulatorPrivateCtx>; |
| **Header file** | #include "ara/crypto/cryp/key\_decapsulator\_private\_ctx.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00301] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00301 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_21501 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::KeyDerivationFunctionCtx |
| **Derived from** | std::unique\_ptr<KeyDerivationFunctionCtx> |
| **Syntax** | using Uptr = std::unique\_ptr<KeyDerivationFunctionCtx>; |
| **Header file** | #include "ara/crypto/cryp/key\_derivation\_function\_ctx.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00302] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00302 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_21801 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::KeyEncapsulatorPublicCtx |
| **Derived from** | std::unique\_ptr<KeyEncapsulatorPublicCtx> |
| **Syntax** | using Uptr = std::unique\_ptr<KeyEncapsulatorPublicCtx>; |
| **Header file** | #include "ara/crypto/cryp/key\_encapsulator\_public\_ctx.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00303] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00303 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_22101 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::MessageAuthnCodeCtx |
| **Derived from** | std::unique\_ptr<MessageAuthnCodeCtx> |
| **Syntax** | using Uptr = std::unique\_ptr<MessageAuthnCodeCtx>; |
| **Header file** | #include "ara/crypto/cryp/message\_authn\_code\_ctx.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00304] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00304 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_22201 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::MsgRecoveryPublicCtx |
| **Derived from** | std::unique\_ptr<MsgRecoveryPublicCtx> |
| **Syntax** | using Uptr = std::unique\_ptr<MsgRecoveryPublicCtx>; |
| **Header file** | #include "ara/crypto/cryp/msg\_recovery\_public\_ctx.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00305] Uptrc

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00305 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_22501 |
| **CR** | Uptrc |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptrc |
| **Scope** | class ara::crypto::cryp::PrivateKey |
| **Derived from** | std::unique\_ptr<const PrivateKey> |
| **Syntax** | using Uptrc = std::unique\_ptr<const PrivateKey>; |
| **Header file** | #include "ara/crypto/cryp/cryobj/private\_key.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** |  |

#### [SWRD-API-Crypto-00306] Uptrc

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00306 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_22701 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptrc |
| **Scope** | class ara::crypto::cryp::PublicKey |
| **Derived from** | std::unique\_ptr<const PublicKey> |
| **Syntax** | using Uptrc = std::unique\_ptr<const PublicKey>; |
| **Header file** | #include "ara/crypto/cryp/cryobj/public\_key.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00307] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-ID (SWRD-ID 编号规则参见附录A-信息定义需求ID。) |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_22901 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::RandomGeneratorCtx |
| **Derived from** | std::unique\_ptr<RandomGeneratorCtx> |
| **Syntax** | using Uptr = std::unique\_ptr<RandomGeneratorCtx>; |
| **Header file** | #include "ara/crypto/cryp/random\_generator\_ctx.h" |
| **Description** | 接口共享智能指针。 |
| **Additional** |  |

#### [SWRD-API-Crypto-00308] Usage

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00308 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_24801 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Usage |
| **Scope** | class ara::crypto::cryp::RestrictedUseObject |
| **Derived from** | AllowedUsageFlags |
| **Syntax** | using Usage = AllowedUsageFlags; |
| **Header file** | #include "ara/crypto/cryp/cryobj/restricted\_use\_object.h" |
| **Description** | 对象允许使用的位标志的容器类型的别名。 |
| **Additional** | - |

#### [SWRD-API-Crypto-00309] Uptrc

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00309 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23001 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptrc |
| **Scope** | class ara::crypto::cryp::SecretSeed |
| **Derived from** | std::unique\_ptr<const SecretSeed> |
| **Syntax** | using Uptrc = std::unique\_ptr<const SecretSeed>; |
| **Header file** | #include "ara/crypto/cryp/cryobj/secret\_seed.h" |
| **Description** | 恒定接口实例的唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00310] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00310 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23002 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::SecretSeed |
| **Derived from** | std::unique\_ptr<SecretSeed> |
| **Syntax** | using Uptr = std::unique\_ptr<SecretSeed>; |
| **Header file** | #include "ara/crypto/cryp/cryobj/secret\_seed.h" |
| **Description** | 接口实例唯一智能指针。 |
| **Additional** | - |

#### [SWRD-API-Crypto-00311] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00311 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23201 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::SigEncodePrivateCtx |
| **Derived from** | std::unique\_ptr<SigEncodePrivateCtx> |
| **Syntax** | using Uptr = std::unique\_ptr<SigEncodePrivateCtx>; |
| **Header file** | #include "ara/crypto/cryp/sig\_encode\_private\_ctx.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00312] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00312 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_29001 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::SignatureService |
| **Derived from** | std::unique\_ptr<SignatureService> |
| **Syntax** | using Uptr = std::unique\_ptr<SignatureService>; |
| **Header file** | #include "ara/crypto/cryp/signature\_service.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00313] Uptrc

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00313 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23301 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptrc |
| **Scope** | class ara::crypto::cryp::Signature |
| **Derived from** | std::unique\_ptr<const Signature> |
| **Syntax** | using Uptrc = std::unique\_ptr<const Signature>; |
| **Header file** | #include "ara/crypto/cryp/cryobj/signature.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00314] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-ID (SWRD-ID 编号规则参见附录A-信息定义需求ID。) |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23501 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::SignerPrivateCtx |
| **Derived from** | std::unique\_ptr<SignerPrivateCtx> |
| **Syntax** | using Uptr = std::unique\_ptr<SignerPrivateCtx>; |
| **Header file** | #include "ara/crypto/cryp/signer\_private\_ctx.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00315] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00315 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23601 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::StreamCipherCtx |
| **Derived from** | std::unique\_ptr<StreamCipherCtx> |
| **Syntax** | using Uptr = std::unique\_ptr<StreamCipherCtx>; |
| **Header file** | #include "ara/crypto/cryp/stream\_cipher\_ctx.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00316] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00316 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23701 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::SymmetricBlockCipherCtx |
| **Derived from** | std::unique\_ptr<SymmetricBlockCipherCtx> |
| **Syntax** | using Uptr = std::unique\_ptr<SymmetricBlockCipherCtx>; |
| **Header file** | #include "ara/crypto/cryp/symmetric\_block\_cipher\_ctx.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00317] Uptrc

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00317 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23801 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptrc |
| **Scope** | class ara::crypto::cryp::SymmetricKey |
| **Derived from** | std::unique\_ptr<const SymmetricKey> |
| **Syntax** | using Uptrc = std::unique\_ptr<const SymmetricKey>; |
| **Header file** | #include "ara/crypto/cryp/cryobj/symmetric\_key.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00318] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00318 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_24001 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::cryp::SymmetricKeyWrapperCtx |
| **Derived from** | std::unique\_ptr<SymmetricKeyWrapperCtx> |
| **Syntax** | using Uptr = std::unique\_ptr<SymmetricKeyWrapperCtx>; |
| **Header file** | #include "ara/crypto/cryp/symmetric\_key\_wrapper\_ctx.h" |
| **Description** | 接口唯一智能指针 |
| **Additional** | - |

#### [SWRD-API-Crypto-00319] mCOType

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00319 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_20506 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | mCOType |
| **Scope** | struct ara::crypto::cryp::CryptoObject::COIdentifier |
| **Type** | CryptoObjectType |
| **Syntax** | CryptoObjectType mCOType; |
| **Header file** | #include "ara/crypto/cryp/cryobj/crypto\_object.h" |
| **Description** | type of objext |
| **Additional** | - |

#### [SWRD-API-Crypto-00320] kObjectType

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00320 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_22503 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kObjectType |
| **Scope** | class ara::crypto::cryp::PrivateKey |
| **Type** | const CryptoObjectType |
| **Syntax** | static const CryptoObjectType kObjectType = CryptoObjectType::kPrivate Key; |
| **Header file** | #include "ara/crypto/cryp/cryobj/private\_key.h" |
| **Description** | 该接口的静态映射到CryptoObjectType枚举的特定值。 |
| **Additional** | - |

#### [SWRD-API-Crypto-00321] kObjectType

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00321 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_22702 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kObjectType |
| **Scope** | class ara::crypto::cryp::PublicKey |
| **Type** | const CryptoObjectType |
| **Syntax** | static const CryptoObjectType kObjectType = CryptoObjectType::kPublic Key; |
| **Header file** | #include "ara/crypto/cryp/cryobj/public\_key.h" |
| **Description** | const object type |
| **Additional** | - |

#### [SWRD-API-Crypto-00322] kObjectType

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00322 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23003 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kObjectType |
| **Scope** | class ara::crypto::cryp::SecretSeed |
| **Type** | const CryptoObjectType |
| **Syntax** | static const CryptoObjectType kObjectType = CryptoObjectType::kSecret Seed; |
| **Header file** | #include "ara/crypto/cryp/cryobj/secret\_seed.h" |
| **Description** | 该接口的静态映射到CryptoObjectType枚举的特定值。 |
| **Additional** | - |

#### [SWRD-API-Crypto-00323] kObjectType

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00323 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23302 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kObjectType |
| **Scope** | class ara::crypto::cryp::Signature |
| **Type** | const CryptoObjectType |
| **Syntax** | static const CryptoObjectType kObjectType = CryptoObjectType::k Signature; |
| **Header file** | #include "ara/crypto/cryp/cryobj/signature.h" |
| **Description** | 签名对象初始化。 |
| **Additional** | - |

#### [SWRD-API-Crypto-00324] kObjectType

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00324 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_23802 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kObjectType |
| **Scope** | class ara::crypto::cryp::SymmetricKey |
| **Type** | const CryptoObjectType |
| **Syntax** | static const CryptoObjectType kObjectType = CryptoObjectType::k SymmetricKey; |
| **Header file** | #include "ara/crypto/cryp/cryobj/symmetric\_key.h" |
| **Description** | const object type |
| **Additional** | - |

#### [SWRD-API-Crypto-00325] mGeneratorUid

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00325 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10101 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | mGeneratorUid |
| **Scope** | struct ara::crypto::CryptoObjectUid |
| **Type** | Uuid |
| **Syntax** | Uuid mGeneratorUid; |
| **Header file** | #include "ara/crypto/common/crypto\_object\_uid.h" |
| **Description** | 生成此COUID的生成器的UUID。该UUID可以与HSM、物理主机/ECU或虚拟机关联。 |
| **Additional** | - |

#### [SWRD-API-Crypto-00102] GenerateSymmetricKey

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-00102 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20725 SWS\_CRYPT\_04207 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetPayloadStorageSize(CryptoObjectType cryptoObjectType, AlgId algId) | |
| **Scope** | class ara::crypto::cryp::CryptoProvider | |
| **Syntax** | virtual ara::core::Result<std::size\_t> GetPayloadStorageSize (Crypto ObjectType cryptoObjectType, AlgId algId) const noexcept=0; | |
| **Parameters (in)** | cryptoObjectType | 目标对象类型 |
| algId | 目标对象的算法ID |
| **Return value** | ara::core::Result< std::size\_t > | 在可信容器中存储的对象的最小空间大小 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUnknown Identifier | 如果参数不支持 |
| CryptoErrorDomain::kIncompatible Arguments | 如果参数不匹配 |
| **Header file** | #include "ara/crypto/cryp/crypto\_provider.h" | |
| **Description** | 返回保存对象有效载荷所需的密钥槽的最小容量。返回值不考虑对象的元信息属性，但它们的大小是固定的，并且对于独立于其实际类型的所有加密对象都是通用的。在TrustedContainer的分配过程中，加密提供商(和密钥存储提供商)根据其实现细节自动为对象的元信息预留空间。 | |
| **Additional** | - | |

### Key Storage Provider

#### [SWRD-API-Crypto-02001] KeySlot

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02001 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30400 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | class |
| **Symbol** | KeySlot |
| **Scope** | namespace ara::crypto::keys |
| **Syntax** | class KeySlot {...}; |
| **Header file** | #include "ara/crypto/keys/keyslot.h" |
| **Description** | KeySlot端口原型接口。 此类允许访问物理KeySlot。 |
| **Additional** |  |

#### [SWRD-API-Crypto-02002] KeyStorageProvider

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02002 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30100 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | class |
| **Symbol** | KeyStorageProvider |
| **Scope** | namespace ara::crypto::keys |
| **Syntax** | class KeyStorageProvider {...}; |
| **Header file** | #include "ara/crypto/keys/key\_storage\_provider.h" |
| **Description** | 密钥存储提供程序接口。 任何对象都由其 UUID 和类型的组合唯一标识。 HSM/TPM 应该使用自己的外部提供的加密对象的副本, 以实现“不可提取密钥”概念。 如果支持相同的格式，一些软件 Crypto Provider 可以共享单个密钥槽。 |

#### [SWRD-API-Crypto-02003] UpdatesObserver

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02003 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30200 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | class |
| **Symbol** | UpdatesObserver |
| **Scope** | namespace ara::crypto::keys |
| **Syntax** | class UpdatesObserver {...}; |
| **Header file** | #include "ara/crypto/keys/updates\_observer.h" |
| **Description** | “更新观察者”接口的定义。 |
| **Additional** | 如果软件开发人员想获得有关插槽内容更新事件的通知，则“更新观察者”接口应由消费者应用程序实现。 |

#### [SWRD-API-Crypto-02004] Clear()

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02004 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30405 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Clear() | |
| **Scope** | class ara::crypto::keys::KeySlot | |
| **Syntax** | virtual ara::core::Result<void> Clear () noexcept=0; | |
| **Thread Safety** | Thread-safe | |
| **Exception Safety** | noexcept | |
| **Parameters (in)** | - | - |
| **Return value** | ara::core::Result< void > | - |
| **Errors:** | CryptoErrorDomain::kUnreserved Resource | 目标槽未打开可写。 |
| **Header file** | #include "ara/crypto/keys/keyslot.h" | |
| **Description** | 清除此KeySlot的内容。 | |
| **Additional** | 此方法必须执行安全清理，但无法恢复对象数据！  此方法可用于对某个范围内的密钥槽进行原子更新。  在这种情况下，只有在对应调用 CommitTransaction() 之后才会更新插槽。 | |

#### [SWRD-API-Crypto-02005] KeySlotContentProps()

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02005 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30510 |
| **CR** | - |
| **Consistency** | Yes/No，Yes:与AutoSar标准一致；No:非标 |
| **Change Type** | 新增/修改/删除/新增 |
| ***Kind*** | function |
| **Symbol** | KeySlotContentProps() |
| **Scope** | struct ara::crypto::keys::KeySlotContentProps |
| **Syntax** | KeySlotContentProps ()=default; |
| **Header file** | #include ara/crypto/keys/key\_slot\_content\_props.h" |
| **Description** | 设置内容属性 |

#### [SWRD-API-Crypto-02006] ~KeySlot()

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02006 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30401 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ~KeySlot() | |
| **Scope** | class ara::crypto::keys::KeySlot | |
| **Syntax** | virtual ~KeySlot () noexcept=default; | |
| **Exception Safety** | noexcept | |
| **Parameters (in)** |  |  |
| **Return value** |  |  |
| **Errors:** |  |  |
| **Header file** | #include "ara/crypto/keys/keyslot.h" | |
| **Description** | 析构函数。 | |

#### [SWRD-API-Crypto-02007] GetContentProps()

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02007 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30408 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetContentProps() | |
| **Scope** | class ara::crypto::keys::KeySlot | |
| **Syntax** | virtual ara::core::Result<KeySlotContentProps> GetContentProps ()  const noexcept=0; | |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Parameters (in)** |  |  |
| **Return value** | ara::core::Result< KeySlotContentProps > |  |
| **Errors** | CryptoErrorDomain::kEmptyContainer | 槽为空槽 |
|  | CryptoErrorDomain::kAccessViolation | 对密钥槽没有任何（“所有者”或“用户”）访问权限 |
| **Header file** | #include "ara/crypto/keys/keyslot.h" | |
| **Description** | 获取KeySlot中内容的实际属性。 如果此方法由“用户”Actor 调用，则始终：props.exportability == false。 | |

#### [SWRD-API-Crypto-02008] MyProvider()

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02008 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30403 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | MyProvider() | |
| **Scope** | class ara::crypto::keys::KeySlot | |
| **Syntax** | virtual ara::core::Result<cryp::CryptoProvider::Uptr> MyProvider ()  const noexcept=0; | |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Parameters (in)** |  |  |
| **Return value** | ara::core::Result< cryp::Crypto  Provider::Uptr > | 要与此密钥槽一起使用的 CryptoProvider 的 unique\_ptr |
| **Header file** | #include "ara/crypto/keys/keyslot.h" | |
| **Description** | 检索拥有此 KeySlot 的 CryptoProvider 实例。  任何密钥槽始终具有关联的默认Crypto Provider的特征，进而Crypto Provider可以为该密钥槽提供服务。  在最简单的情况下，所有密钥槽都可以由安装在自适应平台上的单个 Crypto Provider 提供服务。  但在更复杂的情况下，系统中可能会共存一些不同的 Crypto Provider，例如，如果 ECU 也有一个或几个 HSM 和软件加密实现，并且每个都有自己的物理密钥存储。 在这种情况下，不同的专用Crypto Provider可以由HSM 和软件实现提供服务。 | |

#### [SWRD-API-Crypto-02009] GetPrototypedProps()

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02009 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30407 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetPrototypedProps() | |
| **Scope** | class ara::crypto::keys::KeySlot | |
| **Syntax** | virtual ara::core::Result<KeySlotPrototypeProps> GetPrototypedProps ()const noexcept=0; | |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Parameters (in)** |  |  |
| **Return value** | ara::core::Result< KeySlotPrototypeProps > | 要与此密钥槽一起使用的 CryptoProvider 的 unique\_ptr |
| **Header file** | #include "ara/crypto/keys/keyslot.h" | |
| **Description** | 获取KeySlot的原型属性。 | |

#### [SWRD-API-Crypto-02010] IsEmpty()

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02010 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30404 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | IsEmpty() | |
| **Scope** | class ara::crypto::keys::KeySlot | |
| **Syntax** | virtual bool IsEmpty () const noexcept=0; | |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Parameters (in)** |  |  |
| **Return value** | bool | 如果插槽为空，则为 true，否则为 false |
| **Header file** | #include "ara/crypto/keys/keyslot.h" | |
| **Description** | 检查插槽是否为空。 | |

#### [SWRD-API-Crypto-02011] Open

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02011 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30409 | |
| **CR** | - | |
| **Consistency** | Ye | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Open(bool subscribeForUpdates=false, bool writeable=false) | |
| **Scope** | class ara::crypto::keys::KeySlot | |
| **Syntax** | virtual ara::core::Result<IOInterface::Uptr> Open (bool subscribeFor  Updates=false, bool writeable=false) const noexcept=0; | |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Parameters (in)** | subscribeForUpdates | if this flag is true then the UpdatesObserver instance (previously registered by a call of the method RegisterObserver()) will be subscribed for updates of the opened key slot  如果此标志为真，则 UpdatesObserver 实例（之前通过调用 RegisterObserver() 方法注册）将订阅打开的密钥槽的更新 |
| writeable | indicates whether the key-slot shall be opened  read-only (default) or with write accessd  指示是否应打开钥匙槽  只读（默认）或具有写访问权限 |
| **Return value** | ara::core::Result< IOInterface::Uptr > | 指向 IOInterface 的唯一智能指针  与插槽内容相关联 |
| **Errors** | CryptoErrorDomain::kInvalidUsage Order | subscribeForUpdate为true，但在 Key Storage Provider 上下文中没有注册的 UpdatesObserver 实例 |
|  | CryptoErrorDomain::kBusyResource | 如果指定的槽 writeable 为 true，忙碌，但是 (a) KeySlot已经打开可写，和/或 (b) KeySlot在另一个正在进行的事务的范围内 |
|  | CryptoErrorDomain::kModifiedResource | KeySlot 打开后指定的 slot 被修改 |
| **Header file** | #include "ara/crypto/keys/keyslot.h" | |
| **Description** | 打开这个KeySlot并返回一个 IOInterface。 如果为 RegisterObserver() 的调用提供了 UpdatesObserver 接口，则每次更新此插槽时（并且对“用户”可见）都应由密钥存储引擎（在专用线程中）调用 UpdatesObserver::OnUpdate() 方法 。 即使在销毁返回的 TrustedContainer 之后，仍将继续监视打开的密钥槽，因为槽的内容可能会加载到易失性内存（作为 CryptoObject 或加密原语的 CryptoContext），但在此之后，TrustedContainer 可能会被销毁 . 因此，如果您需要终止对 key slot 的监控，那么您应该直接调用方法 UnsubscribeObserver(SlotNumber)。 | |

#### [SWRD-API-Crypto-02012] KeySlotPrototypeProps()

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02012 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30301 |
| **CR** | - |
| **Consistency** | Yes/No，Yes:与AutoSar标准一致；No:非标 |
| **Change Type** | 新增/修改/删除/新增 |
| ***Kind*** | function |
| **Symbol** | KeySlotPrototypeProps() |
| **Scope** | struct ara::crypto::keys::KeySlotPrototypeProps |
| **Syntax** | KeySlotPrototypeProps ()=default; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_prototype\_props.h" |
| **Description** | 槽号属性设置 |

#### [SWRD-API-Crypto-02013] SaveCopy

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02013 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30406 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | SaveCopy(const IOInterface &container) | |
| **Scope** | class ara::crypto::keys::KeySlot | |
| **Syntax** | virtual ara::core::Result<void> SaveCopy (const IOInterface  &container) noexcept=0; | |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Parameters (in)** | container | the source IOInterface  源 IOInterface |
| **Return value** | ara::core::Result< void > | true if successfully saved  保持成功 |
| **Errors** | CryptoErrorDomain::kIncompatibleObject | 源对象具有属性“会话”或者源 IOInterface 引用来自不同 CryptoProvider 的 KeySlot |
| CryptoErrorDomain::kEmptyContainer | 源 IOInterface 为空 |
| CryptoErrorDomain::kContentRestrictions | 源对象不满足插槽限制（包括版本控制） |
| CryptoErrorDomain::kUnreserved Resource | 目标插槽未打开可写。 |
| **Header file** | #include "ara/crypto/keys/keyslot.h" | |
| **Description** | 将提供的源 IOInterface 的内容保存到此KeySlot。 源容器可能代表一个易失的可信容器或另一个 KeySlot。 此方法可用于对某个事务范围内的密钥槽进行原子更新。 在这种情况下，只有在对应调用 CommitTransaction() 之后才会更新插槽。 | |

#### [SWRD-API-Crypto-02014] operator=

|  |  |  |  |
| --- | --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02014 | | |
| **Type** | Valid | | |
| **Priority** | H | | |
| **Upstream ID** | SWS\_CRYPT\_30220 | | |
| **CR** | - | | |
| **Consistency** | Yes | | |
| **Change Type** | 新增 | | |
| ***Kind*** | function | | |
| **Symbol** | operator=(const KeySlot &other) | | |
| **Scope** | class ara::crypto::keys::KeySlot | | |
| **Syntax** | KeySlot& operator= (const KeySlot &other)=default; | | |
| **Parameters (in)** | other | 其它 instance | |
| **Return value** | KeySlot & | | \*this，包含其它内容 |
| **Errors** |  | | |
| **Header file** | #include "ara/crypto/keys/keyslot.h" | | |
| **Description** | 将另一个 KeySlot 复制分配给此实例。 | | |

#### [SWRD-API-Crypto-02015] operator=

|  |  |  |  |
| --- | --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02015 | | |
| **Type** | Valid | | |
| **Priority** | H | | |
| **Upstream ID** | SWS\_CRYPT\_30221 | | |
| **CR** | - | | |
| **Consistency** | Yes | | |
| **Change Type** | 新增 | | |
| ***Kind*** | function | | |
| **Symbol** | operator=(KeySlot &&other) | | |
| **Scope** | class ara::crypto::keys::KeySlot | | |
| **Syntax** | KeySlot& operator= (KeySlot &&other)=default; | | |
| **Parameters (in)** | other | 其它 instance | |
| **Return value** | KeySlot & | | \*this，包含其它内容 |
| **Header file** | #include "ara/crypto/keys/keyslot.h" | | |
| **Description** | 将另一个 KeySlot 移动分配给该实例。 | | |

#### [SWRD-API-Crypto-02016] BeginTransaction

|  |  |  |  |
| --- | --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02016 | | |
| **Type** | Valid | | |
| **Priority** | H | | |
| **Upstream ID** | SWS\_CRYPT\_30123 | | |
| **CR** | - | | |
| **Consistency** | Yes | | |
| **Change Type** | 新增 | | |
| ***Kind*** | function | | |
| **Symbol** | BeginTransaction(const TransactionScope &targetSlots) | | |
| **Scope** | class ara::crypto::keys::KeyStorageProvider | | |
| **Syntax** | virtual ara::core::Result<TransactionId> BeginTransaction (const  TransactionScope &targetSlots) noexcept=0; | | |
| **Parameters (in)** | targetSlots | 在此事务期间应更新的 KeySlot 列表。 | |
| **Return value** | ara::core::Result< TransactionId > | | 分配给此交易的唯一 ID |
| **Exception Safety** | noexcept | | |
| **Thread Safety** | Thread-safe | | |
| **Errors** | CryptoErrorDomain::kUnreservedResource | | targetSlots 列表中有一个未在清单中使用 reserveSpareSlot 参数配置的槽号 |
| CryptoErrorDomain::kBusyResource | | targetSlots 列表具有已涉及另一个待处理事务或以写入模式打开的关KeySlot |
| **Header file** | #include "ara/crypto/keys/key\_storage\_provider.h" | | |
| **Description** | 开始新的槽号列表更新事务。 | | |
| **Additional** | 1. KeySlot的 reserveSpareSlot 模型参数必须设置为 true，keyslot才能作为事务范围被处理。 2. 事务专用于同时更新相关的密钥槽（以原子的、全有或全无的方式）。 3. 所有应该由事务更新的密钥槽都必须打开并提供给这个函数。 4. 通过调用 commit() 执行对范围内插槽的任何更改。 | | |

#### [SWRD-API-Crypto-02017] CommitTransaction

|  |  |  |  |
| --- | --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02017 | | |
| **Type** | Valid | | |
| **Priority** | H | | |
| **Upstream ID** | SWS\_CRYPT\_30124 | | |
| **CR** | - | | |
| **Consistency** | Yes | | |
| **Change Type** | 新增 | | |
| ***Kind*** | function | | |
| **Symbol** | CommitTransaction(TransactionId id) | | |
| **Scope** | class ara::crypto::keys::KeyStorageProvider | | |
| **Syntax** | virtual ara::core::Result<void> CommitTransaction (TransactionId id)  noexcept=0; | | |
| **Parameters (in)** | id | 应提交的事务的 ID。 | |
| **Return value** | ara::core::Result< void > |  | |
| **Exception Safety** | noexcept | | |
| **Thread Safety** | Thread-safe | | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | | 提供的 ID 无效，即此 ID 未知或对应事务已完成（提交或回滚） |
| **Header file** | #include "ara/crypto/keys/key\_storage\_provider.h" | | |
| **Description** | 将事务的更改提交到密钥存储。 | | |
| **Additional** | 1.在事务期间对密钥槽所做的任何更改在提交执行之前都是不可见的。 2.提交命令将事务期间所做的所有更改永久保存在密钥存储中。 | | |

#### [SWRD-API-Crypto-02018] ~KeyStorageProvider()

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02018 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30110 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | function |
| **Symbol** | ~KeyStorageProvider() |
| **Scope** | class ara::crypto::keys::KeyStorageProvider |
| **Syntax** | virtual ~KeyStorageProvider () noexcept=default; |
| **Exception Safety** | noexcept |
| **Header file** | #include "ara/crypto/keys/key\_storage\_provider.h" |
| **Description** | 析构函数 |

#### [SWRD-API-Crypto-02019] GetRegisteredObserver()

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02019 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30110 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetRegisteredObserver() | |
| **Scope** | class ara::crypto::keys::KeyStorageProvider | |
| **Syntax** | virtual UpdatesObserver::Uptr GetRegisteredObserver () const  noexcept=0; | |
| **Return value** | UpdatesObserver::Uptr | 指向已注册更新观察者接口的唯一指针（返回副本的内部unique指针，即Key Storage provider继续保留所有权） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/keys/key\_storage\_provider.h" | |
| **Description** | 获取已注册更新观察者的指针。 如果尚未注册观察者，该方法将返回 nullptr！ | |

#### [SWRD-API-Crypto-02020] LoadKeySlot

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02020 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30115 | |
| **CR** | - | |
| **Consistency** | Yes/No，Yes:与AutoSar标准一致；No:非标 | |
| **Change Type** | 新增/修改/删除/新增 | |
| ***Kind*** | function | |
| **Symbol** | LoadKeySlot(ara::core::InstanceSpecifier &iSpecify) | |
| **Scope** | class ara::crypto::keys::KeyStorageProvider | |
| **Syntax** | virtual ara::core::Result<KeySlot::Uptr> LoadKeySlot (ara::core::InstanceSpecifier &iSpecify) noexcept=0; | |
| **Parameters (in)** | iSpecify | 目标槽实例说明符 |
| **Return value** | ara::core::Result< KeySlot::Uptr > | 指向已分配密钥槽的智能指针 |
| **Errors** | CryptoErrorDomain::kUnreserved Resource | InstanceSpecifier 不正确（未分配插槽） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/keys/key\_storage\_provider.h" | |
| **Description** | 加载一个密钥槽。 | |
| **Additional** | 将与 KeySlot 关联的信息加载到 KeySlot 对象中。 | |

#### [SWRD-API-Crypto-02021] RegisterObserver

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02021 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30130 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | RegisterObserver(UpdatesObserver::Uptr observer=nullptr) | |
| **Scope** | class ara::crypto::keys::KeyStorageProvider | |
| **Syntax** | virtual UpdatesObserver::Uptr RegisterObserver (UpdatesObserver::Uptr observer=nullptr) noexcept=0; | |
| **Parameters (in)** | observer | 指向客户端提供的更新观察者实例的可选指针，该实例应在密钥存储实现中注册并每次调用，当一个打开的使用/加载密钥槽在外部更新时（通过其“所有者”应用程序） |
| **Return value** | UpdatesObserver::Uptr | 指向先前注册的更新观察者接口的唯一指针（指针所有权被“移出”到调用者代码） |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/keys/key\_storage\_provider.h" | |
| **Description** | 注册消费者更新观察者。 | |
| **Additional** | 一个应用程序进程只能注册一个 UpdatesObserver 实例，因此该方法总是注销前一个观察者并返回其唯一指针。 如果 (nullptr == observer) 那么该方法只会注销前一个观察者！ 如果尚未注册观察者，该方法将返回 nullptr！ | |

#### [SWRD-API-Crypto-02022] RollbackTransaction

|  |  |  |  |
| --- | --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02022 | | |
| **Type** | Valid | | |
| **Priority** | H | | |
| **Upstream ID** | SWS\_CRYPT\_30124 | | |
| **CR** | - | | |
| **Consistency** | Yes | | |
| **Change Type** | 新增 | | |
| ***Kind*** | function | | |
| **Symbol** | RollbackTransaction(TransactionId id) | | |
| **Scope** | class ara::crypto::keys::KeyStorageProvider | | |
| **Syntax** | virtual ara::core::Result<void> RollbackTransaction (TransactionId id)  noexcept=0; | | |
| **Parameters (in)** | id | 应回滚的事务的 ID。 | |
| **Return value** | ara::core::Result< void > | *-* | |
| **Exception Safety** | noexcept | | |
| **Thread Safety** | Thread-safe | | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | | 提供的 ID 无效，即此 ID 未知或对应事务已完成（提交或回滚） |
| **Header file** | #include "ara/crypto/keys/key\_storage\_provider.h" | | |
| **Description** | 回滚密钥存储中事务期间执行的所有更改。 | | |
| **Additional** | 1. 回滚命令永久取消在密钥存储中的事务期间所做的所有更改。 2. 回滚事务对所有应用程序都是完全不可见的 | | |

#### [SWRD-API-Crypto-02023] UnsubscribeObserver

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02023 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30126 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | UnsubscribeObserver(KeySlot &slot) | |
| **Scope** | class ara::crypto::keys::KeyStorageProvider | |
| **Syntax** | virtual ara::core::Result UnsubscribeObserver (KeySlot &slot) noexcept=0; | |
| **Parameters (in)** | slot | 从更新观察中取消订阅的槽号 |
| **Return value** | ara::core::Result< void > | *-* |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 指定的插槽现在没有被监控（即如果它没有通过 OpenAsUser() 成功打开或者它已经被这个方法取消订阅） |
| **Header file** | #include "ara/crypto/keys/key\_storage\_provider.h" | |
| **Description** | 从指定槽的更改监视中取消订阅更新观察者。 | |

#### [SWRD-API-Crypto-02024] operator=

|  |  |  |  |
| --- | --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02024 | | |
| **Type** | Valid | | |
| **Priority** | H | | |
| **Upstream ID** | SWS\_CRYPT\_30222 | | |
| **CR** | - | | |
| **Consistency** | Yes | | |
| **Change Type** | 新增 | | |
| ***Kind*** | function | | |
| **Symbol** | operator=(const KeyStorageProvider &other) | | |
| **Scope** | class ara::crypto::keys::KeyStorageProvider | | |
| **Syntax** | KeyStorageProvider& operator= (const KeyStorageProvider &other)=default; | | |
| **Parameters (in)** | other | 其它 instance | |
| **Return value** | KeyStorageProvider & | | \*this，包含其它内容 |
| **Header file** | #include "ara/crypto/keys/key\_storage\_provider.h" | | |
| **Description** | 将另一个 KeyStorageProvider 复制分配给此实例 | | |

#### [SWRD-API-Crypto-02025] operator=

|  |  |  |  |
| --- | --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02025 | | |
| **Type** | Valid | | |
| **Priority** | H | | |
| **Upstream ID** | SWS\_CRYPT\_30223 | | |
| **CR** | - | | |
| **Consistency** | Yes/No，Yes:与AutoSar标准一致；No:非标 | | |
| **Change Type** | 新增/修改/删除/新增 | | |
| ***Kind*** | function | | |
| **Symbol** | operator=(KeyStorageProvider &&other) | | |
| **Scope** | class ara::crypto::keys::KeyStorageProvider | | |
| **Syntax** | KeyStorageProvider& operator= (KeyStorageProvider &&other)=default; | | |
| **Parameters (in)** | other | 其它 instance | |
| **Return value** | KeyStorageProvider & | | \*this，包含其它内容 |
| **Header file** | #include "ara/crypto/keys/key\_storage\_provider.h" | | |
| **Description** | 将另一个 KeyStorageProvider 移动分配给此实例 | | |

#### [SWRD-API-Crypto-02026] operator==

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02026 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30350 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator==(const KeySlotPrototypeProps &lhs, const KeySlotPrototypeProps &rhs) | |
| **Scope** | namespace ara::crypto::keys | |
| **Syntax** | constexpr bool operator== (const KeySlotPrototypeProps &lhs, const Key SlotPrototypeProps &rhs) noexcept; | |
| **Parameters (in)** | lhs | 左值 |
| rhs | 右值 |
| **Return value** | bool | 左右值相同为true 否则false |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/keys/key\_slot\_prototype\_props.h" | |
| **Description** | KeySlotPrototypeProps 操作数的比较运算符“相等”。 | |

#### [SWRD-API-Crypto-02027] operator!=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02027 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30350 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator!=(const KeySlotPrototypeProps &lhs, const KeySlotPrototypeProps &rhs) | |
| **Scope** | namespace ara::crypto::keys | |
| **Syntax** | constexpr bool operator!= (const KeySlotPrototypeProps &lhs, const Key SlotPrototypeProps &rhs) noexcept; | |
| **Parameters (in)** | lhs | 左值 |
| rhs | 右值 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Return value** | bool | 如果 lhs 的至少一个成员的值不等于 rhs 的对应成员，则为 true，否则为 false |
| **Header file** | #include "ara/crypto/keys/key\_slot\_prototype\_props.h" | |
| **Description** | KeySlotPrototypeProps 操作数的比较运算符“不等于” | |

#### [SWRD-API-Crypto-02028] operator==

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02028 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30550 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator==(const KeySlotContentProps &lhs, const KeySlotContentProps &rhs) | |
| **Scope** | namespace ara::crypto::keys | |
| **Syntax** | constexpr bool operator== (const KeySlotContentProps &lhs, const Key SlotContentProps &rhs) noexcept; | |
| **Parameters (in)** | lhs | 左值 |
| rhs | 右值 |
| **Return value** | bool | 如果所有成员的 lhs 值都等于 rhs，则为 true，否则为 false |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/keys/key\_slot\_content\_props.h" | |
| **Description** | KeySlotContentProps 操作数的比较运算符“相等”。 | |

#### [SWRD-API-Crypto-02029] operator!=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02029 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30551 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator!=(const KeySlotContentProps &lhs, const KeySlotContentProps &rhs) | |
| **Scope** | namespace ara::crypto::keys | |
| **Syntax** | constexpr bool operator!= (const KeySlotContentProps &lhs, const Key SlotContentProps &rhs) noexcept; | |
| **Parameters (in)** | lhs | 左值 |
| rhs | 右值 |
| **Return value** | bool | 如果 lhs 的至少一个成员的值不等于 rhs 的对应成员，则为 true，否则为 false |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/keys/key\_slot\_content\_props.h" | |
| **Description** | KeySlotContentProps 操作数的比较运算符“不等于”。 | |

#### [SWRD-API-Crypto-02030] ~UpdatesObserver()

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02030 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30210 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | function |
| **Symbol** | ~UpdatesObserver() |
| **Scope** | class ara::crypto::keys::UpdatesObserver |
| **Syntax** | virtual ~UpdatesObserver () noexcept=default; |
| **Exception Safety** | noexcept |
| **Header file** | #include "ara/crypto/keys/key\_storage\_provider.h" |
| **Description** | 析构函数 |

#### [SWRD-API-Crypto-02031] OnUpdate

|  |  |  |  |
| --- | --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02031 | | |
| **Type** | Valid | | |
| **Priority** | H | | |
| **Upstream ID** | SWS\_CRYPT\_30211 | | |
| **CR** | - | | |
| **Consistency** | Yes | | |
| **Change Type** | 新增 | | |
| ***Kind*** | function | | |
| **Symbol** | OnUpdate(const TransactionScope &updatedSlots) | | |
| **Scope** | class ara::crypto::keys::UpdatesObserver | | |
| **Syntax** | virtual void OnUpdate (const TransactionScope &updatedSlots) noexcept=0; | | |
| **Parameters (in)** | updatedSlots | 打开后更新的监控槽列表（供阅读） | |
| **Return value** | None | | 分配给此交易的唯一 ID |
| **Exception Safety** | noexcept | | |
| **Thread Safety** | Thread-safe | | |
| **Header file** | #include "ara/crypto/keys/updates\_observer.h" | | |
| **Description** | 如果指定插槽的内容发生更改，则应调用的通知方法。 | | |
| **Additional** | 密钥存储引擎应在专用线程中调用此方法。 提供的列表可能仅包括订阅观察的槽（在使用“用户”权限打开期间，即通过调用方法 OpenAsUser() 进行“读取”）。 每个插槽号只能出现在提供的列表中一次！ | | |

#### [SWRD-API-Crypto-02032] operator=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02032 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30224 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator=(const UpdatesObserver &other) | |
| **Scope** | class ara::crypto::keys::UpdatesObserver | |
| **Syntax** | UpdatesObserver& operator= (const UpdatesObserver &other)=default; | |
| **Parameters (in)** | other | 其它 instance |
| **Return value** | UpdatesObserver & | \*this，包含其它内容 |
| **Header file** | #include "ara/crypto/keys/updates\_observer.h" | |
| **Description** | 将另一个 UpdatesObserver 复制分配给此实例 | |

#### [SWRD-API-Crypto-02033] operator=

|  |  |  |  |
| --- | --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02033 | | |
| **Type** | Valid | | |
| **Priority** | H | | |
| **Upstream ID** | SWS\_CRYPT\_30224 | | |
| **CR** | - | | |
| **Consistency** | Yes | | |
| **Change Type** | 新增 | | |
| ***Kind*** | function | | |
| **Symbol** | operator=(UpdatesObserver &&other) | | |
| **Scope** | class ara::crypto::keys::UpdatesObserver | | |
| **Syntax** | UpdatesObserver& operator= (UpdatesObserver &&other)=default; | | |
| **Parameters (in)** | other | 其它 instance | |
| **Return value** | UpdatesObserver & | | \*this，包含其它内容 |
| **Header file** | #include "ara/crypto/keys/updates\_observer.h" | | |
| **Description** | 将另一个 UpdatesObserver 移动分配给该实例。 | | |

#### [SWRD-API-Crypto-02034] KeySlotContentProps

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02034 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30500 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | struct |
| **Symbol** | KeySlotContentProps |
| **Scope** | namespace ara::crypto::keys |
| **Syntax** | struct KeySlotContentProps {...}; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_content\_props.h" |
| **Description** | 当前 Key Slot 内容的属性，即存储到 Key Slot 的当前实例的属性。 mAllowedUsage 字段的值是在运行时定义的公共使用标志和由当前“参与者”的 UserPermissions 原型定义的使用标志的按位与。 |

#### [SWRD-API-Crypto-02035] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02035 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30511 |
| **CR** |  |
| **Consistency** |  |
| **Change Type** | 新增 |
| **Kind** | type alias |
| **Symbol** | Uptr |
| **Scope** | struct ara::crypto::keys::KeySlotContentProps |
| **Derived from** | std::unique\_ptr |
| **Syntax** | using Uptr = std::unique\_ptr; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_content\_props.h" |
| **Description** | 接口共享指针 |

#### [SWRD-API-Crypto-02036] KeySlotPrototypeProps

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02036 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30300 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | struct |
| **Symbol** | KeySlotPrototypeProps |
| **Scope** | namespace ara::crypto::keys |
| **Syntax** | struct KeySlotPrototypeProps {...}; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_prototype\_props.h" |
| **Description** | Key的槽号属性。 |

#### [SWRD-API-Crypto-02037] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02037 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30302 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | type alias |
| **Symbol** | Uptr |
| **Scope** | struct ara::crypto::keys::KeySlotPrototypeProps |
| **Derived from** | std::unique\_ptr |
| **Syntax** | using Uptr = std::unique\_ptr<KeySlotPrototypeProps>; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_prototype\_props.h" |
| **Description** | 接口共享指针 |

#### [SWRD-API-Crypto-02038] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02038 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30402 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::keys::KeySlot |
| **Derived from** | std::unique\_ptr<KeySlot> |
| **Syntax** | using Uptr = std::unique\_ptr<KeySlot>; |
| **Header file** | #include "ara/crypto/keys/keyslot.h" |
| **Description** | 接口的唯一智能指针。 |

#### [SWRD-API-Crypto-02039] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02039 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30101 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::keys::KeyStorageProvider |
| **Derived from** | std::unique\_ptr<KeyStorageProvider> |
| **Syntax** | using Uptr = std::unique\_ptr<KeyStorageProvider>; |
| **Header file** | #include "ara/crypto/keys/key\_storage\_provider.h" |
| **Description** | 接口共享指针 |

#### [SWRD-API-Crypto-02040] TransactionId

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02040 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30010 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | type alias |
| **Symbol** | TransactionId |
| **Scope** | namespace ara::crypto::keys |
| **Derived from** | std::uint64\_t |
| **Syntax** | using TransactionId = std::uint64\_t; |
| **Header file** | #include "ara/crypto/keys/elementary\_types.h" |
| **Description** | 事务标识符类型的定义。 零值应为特殊情况保留。 |

#### [SWRD-API-Crypto-02041] TransactionScope

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02041 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30011 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | type alias |
| **Symbol** | TransactionScope |
| **Scope** | namespace ara::crypto::keys |
| **Derived from** | ara::core::Vector<KeySlot> |
| **Syntax** | using TransactionScope = ara::core::Vector<KeySlot>; |
| **Header file** | #include "ara/crypto/keys/elementary\_types.h" |
| **Description** | “事务范围”类型的定义。 “事务范围”定义了在事务中作为更新目标的KeySlot列表。 |

#### [SWRD-API-Crypto-02042] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02042 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30201 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::keys::UpdatesObserver |
| **Derived from** | std::unique\_ptr<UpdatesObserver> |
| **Syntax** | using Uptr = std::unique\_ptr<UpdatesObserver>; |
| **Header file** | #include "ara/crypto/keys/updates\_observer.h" |
| **Description** | 接口的共享智能指针*。* |

#### [SWRD-API-Crypto-02043] mAlgId

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02043 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30503 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | variable |
| **Symbol** | mAlgId |
| **Scope** | struct ara::crypto::keys::KeySlotContentProps |
| **Type** | CryptoAlgId |
| **Syntax** | CryptoAlgId mAlgId; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_content\_props.h" |
| **Description** | 存储到插槽的实际对象的加密算法。 |

#### [SWRD-API-Crypto-02044] mObjectSize

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02044 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30505 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | variable |
| **Symbol** | mObjectSize |
| **Scope** | struct ara::crypto::keys::KeySlotContentProps |
| **Type** | std::size\_t |
| **Syntax** | std::size\_t mObjectSize; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_content\_props.h" |
| **Description** | 当前存储到插槽中的对象的实际大小。 |

#### [SWRD-API-Crypto-02045] mObjectType

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02045 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30508 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | variable |
| **Symbol** | mObjectType |
| **Scope** | struct ara::crypto::keys::KeySlotContentProps |
| **Type** | CryptoObjectType |
| **Syntax** | CryptoObjectType mObjectType; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_content\_props.h" |
| **Description** | 存储到插槽的对象的实际类型。 |

#### [SWRD-API-Crypto-02046] mObjectUid

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02046 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30501 |
| **CR** |  |
| **Consistency** |  |
| **Change Type** | 新增 |
| **Kind** | variable |
| **Symbol** | mObjectUid |
| **Scope** | struct ara::crypto::keys::KeySlotContentProps |
| **Type** | CryptoObjectUid |
| **Syntax** | CryptoObjectUid mObjectUid; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_content\_props.h" |
| **Description** | 存储到插槽的加密对象的 UID。 |

#### [SWRD-API-Crypto-02047] mContentAllowedUsage

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02047 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30506 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | variable |
| **Symbol** | mContentAllowedUsage |
| **Scope** | struct ara::crypto::keys::KeySlotContentProps |
| **Type** | AllowedUsageFlags |
| **Syntax** | AllowedUsageFlags mContentAllowedUsage; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_content\_props.h" |
| **escription** | 存储到当前“Actor”插槽的对象的实际使用限制标志 |

#### [SWRD-API-Crypto-02048] mAlgId

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02048 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30306 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | variable |
| **Symbol** | mAlgId |
| **Scope** | struct ara::crypto::keys::KeySlotPrototypeProps |
| **Type** | CryptoAlgId |
| **Syntax** | CryptoAlgId mAlgId; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_prototype\_props.h" |
| **Description** | 加密算法限制（kAlgIdAny 表示没有限制）。 该算法可以部分指定：算法组和算法长度，模式，填充。 |

#### [SWRD-API-Crypto-02049] mAllowContentTypeChange

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02049 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30310 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | variable |
| **Symbol** | mAllowContentTypeChange |
| **Scope** | struct ara::crypto::keys::KeySlotPrototypeProps |
| **Type** | bool |
| **Syntax** | bool mAllowContentTypeChange; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_prototype\_props.h" |
| **Description** | 指示此Key的内容是否可以更改，例如 从存储对称密钥到存储 RSA 密钥 如果设置为 false，则此 KeySlotPrototype Props 的 mObjectType 必须 a) 有效且 b) 不能更改（即只有 mObjectType 的对象可以存储在此密钥槽中）。 |

#### [SWRD-API-Crypto-02050] mContentAllowedUsage

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02050 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30313 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | variable |
| **Symbol** | mContentAllowedUsage |
| **Scope** | struct ara::crypto::keys::KeySlotPrototypeProps |
| **Type** | AllowedUsageFlags |
| **Syntax** | AllowedUsageFlags mContentAllowedUsage; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_prototype\_props.h" |
| **Description** | 表示如何使用内容。此属性的用例如下：  kAllowPrototypedOnly：若要将内容加载到运行实例中，该属性值必须设置kAllowPrototypedOnly（例如，应根据此属性设置symmetricKey 对象的 AllowedUsageFlags）  mMaxUpdatesAllowed==0：内容是在生产过程中提供的，而 AllowedUsageFlags 是 灵活更新此KeySlot时使用此属性建模运行时对象的 AllowedUsageFlags 在稍后从此KeySlot加载时覆盖此属性 。 |

#### [SWRD-API-Crypto-02051] mExportAllowed

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02051 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30312 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | variable |
| **Symbol** | mExportAllowed |
| **Scope** | struct ara::crypto::keys::KeySlotPrototypeProps |
| **Type** | bool |
| **Syntax** | bool mExportAllowed; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_prototype\_props.h" |
| **Description** | 表示是否可以导出KeySlot内容。 |

#### [SWRD-API-Crypto-02052] mMaxUpdateAllowed

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02052 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30311 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | variable |
| **Symbol** | mMaxUpdateAllowed |
| **Scope** | struct ara::crypto::keys::KeySlotPrototypeProps |
| **Type** | std::int32\_t |
| **Syntax** | std::int32\_t mMaxUpdateAllowed; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_prototype\_props.h" |
| **Description** | 指定此KeySlot可以更新多少次，例如：  值 0 表示将在生产期间预设KeySlot内容  值 1 表示KeySlot内容只能更新一次（“OTP”）  负值表示KeySlot内容可以无限更新。 |

#### [SWRD-API-Crypto-02053] mSlotType

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02053 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30305 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | variable |
| **Symbol** | mSlotType |
| **Scope** | struct ara::crypto::keys::KeySlotPrototypeProps |
| **Type** | KeySlotType |
| **Syntax** | KeySlotType mSlotType; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_prototype\_props.h" |
| **Description** | 密钥槽类型配置：  kMachine：AP平台用于提供诸如安全通信、诊断、更新、安全存储等服务的所有密钥槽都应使用该类型。  kApplication：自适应用户应用程序将使用的所有KeySlot都必须使用该类型。  其它：密钥管理器用户应用程序也可以定义 kMachine 密钥槽； 在这种情况下，集成商必须匹配要管理的相应机器密钥槽。 |

#### [SWRD-API-Crypto-02054] mSlotCapacity

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02054 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30307 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | variable |
| **Symbol** | mSlotCapacity |
| **Scope** | struct ara::crypto::keys::KeySlotPrototypeProps |
| **Type** | std::size\_t |
| **Syntax** | std::size\_t mSlotCapacity; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_prototype\_props.h" |
| **Description** | 插槽容量（以字节为单位）。 |

#### [SWRD-API-Crypto-02055] mObjectType

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-02055 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30308 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | variable |
| **Symbol** | mObjectType |
| **Scope** | struct ara::crypto::keys::KeySlotPrototypeProps |
| **Type** | CryptoObjectType |
| **Syntax** | CryptoObjectType mObjectType; |
| **Header file** | #include "ara/crypto/keys/key\_slot\_prototype\_props.h" |
| **Description** | 可以存储槽的对象类型的限制。 如果此字段为CryptoObjectType::kUnknown 则不受类型限制。 |

### X.509 Provider

#### [SWRD-API-Crypto-03001] X509Object

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-03001 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_40900 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | class |
| **Symbol** | X509Object |
| **Scope** | namespace ara::crypto::x509 |
| **Base class** | ara::crypto::Serializable |
| **Syntax** | class X509Object : public Serializable {...}; |
| **Header file** | #include "ara/crypto/x509/x509\_object.h" |
| **Description** | X.509 Provider 创建的所有对象的通用接口。 |

#### [SWRD-API-Crypto-03002] X509Provider

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-03002 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_40600 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | class |
| **Symbol** | X509Provider |
| **Scope** | namespace ara::crypto::x509 |
| **Syntax** | class X509Provider {...}; |
| **Header file** | #include "ara/crypto/x509/x509\_provider.h" |
| **Description** | X.509 提供程序接口。 X.509 Provider 支持两种内部存储：易失性（或会话）和持久性。 Provider创建的所有 X.509 对象都应该具有对其父 X.509 Provider的实际引用。 只有在销毁其所有子对象后才能销毁 X.509 Provider。 此接口的每个创建 X.509 对象的方法都是非持久性的，因为任何此类创建都会增加 X.509 Provider 的引用计数器。 |

#### [SWRD-API-Crypto-03003] MyProvider()

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-03003 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_40911 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | MyProvider() | |
| **Scope** | class ara::crypto::x509::X509Object | |
| **Syntax** | virtual X509Provider& MyProvider () const noexcept=0; | |
| **Return value** | X509Provider & | 对提供此对象的 X.509 Provider 实例的引用 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/x509/x509\_object.h" | |
| **Description** | 获取对此对象的 X.509 Provider 的引用。 | |

#### [SWRD-API-Crypto-03004] LoadCertificate

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-03004 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_40641 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | LoadCertificate(ara::core::InstanceSpecifier &iSpecify) | |
| **Scope** | class ara::crypto::x509::X509Provider | |
| **Syntax** | virtual ara::core::Result LoadCertificate ( ara::core::InstanceSpecifier &iSpecify) noexcept=0; | |
| **Parameters (in)** | iSpecify | 目标证书实例的iSpecifiy |
| **Return value** | ara::core::Result< Certificate::Uptr > | 指向实例化证书的唯一智能指针 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kUnreservedResource | InstanceSpecifier 不正确（找不到证书） |
| **Header file** | #include "ara/crypto/x509/x509\_provider.h" | |
| **Description** | 从持久证书存储中加载证书。 | |

#### [SWRD-API-Crypto-03005] ParseCertChain

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-03005 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_40616 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ParseCertChain(ara::core::Vector< Certificate::Uptr > &outcome, ReadOnlyMemRegion cert Chain, Serializable::FormatId formatId=Serializable::kFormatDefault) | |
| **Scope** | class ara::crypto::x509::X509Provider | |
| **Syntax** | virtual ara::core::Result ParseCertChain (ara::core::Vector< Certificate::Uptr > &outcome, ReadOnlyMemRegion certChain, Serializable::FormatId formatId=Serializable::kFormatDefault) noexcept=0; | |
| **Parameters (in)** | certChain | DER/PEM 编码的证书链（以单个 BLOB 的形式） |
| formatId | 输入格式标识符（kFormatDefault 表示自动检测） |
| **Parameters (out)** | outcome | 导入证书链的输出向量 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInsufficient Capacity | 输出容量小于链中证书的实际数量 |
| CryptoErrorDomain::kInvalidArgument | 无法解析 certChain 参数 |
| CryptoErrorDomain::kUnknownIdentifier | formatId 参数为未知值 |
| **Header file** | #include "ara/crypto/x509/x509\_provider.h" | |
| **Description** | 解析证书链的序列化表示并创建它们的实例。  输出结果向量中的证书将从根 CA 证书（0索引）放置，到最终的最终实体证书（向量的最后使用索引）。 | |

#### [SWRD-API-Crypto-03006] ParseCert

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-03006 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_40614 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ParseCert(ReadOnlyMemRegion cert, Serializable::FormatId formatId=Serializable::kFormat Default) | |
| **Scope** | class ara::crypto::x509::X509Provider | |
| **Syntax** | virtual ara::core::Result ParseCertChain (ara::core::Vector< Certificate::Uptr > &outcome, const ara::core::Vector< ReadOnlyMem Region > &certChain, Serializable::FormatId formatId=Serializable::k FormatDefault) noexcept=0; | |
| **Parameters (in)** | certChain | DER/PEM 编码的证书链（每个证书由输入向量中的单独 BLOB 提供） |
| formatId | 输入格式标识符（kFormatDefault 表示自动检测） |
| **Parameters (out)** | outcome | 导入证书链的输出向量 |
| **Return value** | ara::core::Result< void > | - |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kInvalidArgument | 无法解析 cert 参数 |
| CryptoErrorDomain::kUnknownIdentifier | formatId 参数为未知值 |
| **Header file** | #include "ara/crypto/x509/x509\_provider.h" | |
| **Description** | 解析证书的序列化表示并创建其实例。 | |

#### [SWRD-API-Crypto-03007] VerifyCert

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-03007 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_40618 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | VerifyCert(Certificate &cert, Certificate::Uptr myRoot=nullptr) | |
| **Scope** | class ara::crypto::x509::X509Provider | |
| **Syntax** | virtual Certificate::Status VerifyCert (Certificate &cert, Certificate::Uptr myRoot=nullptr) noexcept=0; | |
| **Parameters (in)** | cert | 用于验证的目标证书 |
| myRoot | 用于验证的根证书 - 如果这是 nullptr，则使用机器根证书 |
| **Return value** | Certificate::Status | 所提供证书的验证状态 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/x509/x509\_provider.h" | |
| **Description** | 通过本地存储的 CA 证书验证所提供证书的状态。 | |

#### [SWRD-API-Crypto-03008] VerifyCertChain

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-03008 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_40619 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | virtual Certificate::Status VerifyCertChain (ara::core::Span< const Certificate::Uptr > chain, Certificate::Uptr myRoot=nullptr) const noexcept=0; | |
| **Scope** | class ara::crypto::x509::X509Provider | |
| **Syntax** | virtual ara::core::Result<bool> UpdateCrlOnline (const Certificate &ca  Cert) noexcept=0; | |
| **Parameters (in)** | chain | 待验证的目标证书链 |
| myRoot | 用于验证的根证书，如果这是 nullptr，则使用机器根证书 |
| **Return value** | Certificate::Status | 提供的证书链的验证状态 |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Header file** | #include "ara/crypto/x509/x509\_provider.h" | |
| **Description** | 通过本地存储的 CA 证书验证证书链的状态。  链中的证书（由容器向量提供）必须从根 CA 证书（0索引）放置到目标终端实体证书（向量的最后使用索引）。 验证以相同的顺序执行。 如果链验证失败，则返回第一个失败证书的状态。 | |

#### [SWRD-API-Crypto-03009] ~X509Provider()

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-03009 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_40604 |
| **CR** | - |
| **Consistency** | Yes/No，Yes:与AutoSar标准一致；No:非标 |
| **Change Type** | 新增/修改/删除/新增 |
| ***Kind*** | function |
| **Symbol** | ~X509Provider() |
| **Scope** | class ara::crypto::x509::X509Provider |
| **Syntax** | virtual ~X509Provider () noexcept=default; |
| **Exception Safety** | noexcept |
| **Header file** | #include "ara/crypto/x509/x509\_provider.h" |
| **Description** | 析构函数。 |

#### [SWRD-API-Crypto-03010] Uptr

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-03010 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_40601 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | type alias |
| **Symbol** | Uptr |
| **Scope** | class ara::crypto::x509::X509Provider |
| **Derived from** | std::unique\_ptr<X509Provider> |
| **Syntax** | using Uptr = std::unique\_ptr<X509Provider>; |
| **Header file** | #include "ara/crypto/x509/x509\_provider.h" |
| **Description** | 接口的共享智能指针。 |

#### [SWRD-API-Crypto-03011] kInvalidIndex

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-03011 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_40603 |
| **CR** |  |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| **Kind** | variable |
| **Symbol** | kInvalidIndex |
| **Scope** | class ara::crypto::x509::X509Provider |
| **Type** | const StorageIndex |
| **Syntax** | static const StorageIndex kInvalidIndex = static\_cast<std::size\_  t>(-1LL); |
| **Header file** | #include "ara/crypto/x509/x509\_provider.h" |
| **Description** | 为证书存储中的导航保留“无效索引”值。 |

### API Reference

#### [SWRD-API-Crypto-04001] IOInterface

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04001 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10800 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | class |
| **Symbol** | IOInterface |
| **Scope** | namespace ara::crypto |
| **Syntax** | class IOInterface {...}; |
| **Header file** | #include "ara/crypto/common/io\_interface.h" |
| **Description** | IOInterface 的正式接口用于保存和加载安全对象。 实际的保存和加载应该通过一对受信任的 Crypto Provider 和 Storage Provider 已知的内部方法来实现。 每个对象都应由其类型和加密对象唯一标识符 (COUID) 唯一标识。 该接口假设容器中的对象被压缩，即具有优化的最小尺寸。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04002] Serializable

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04002 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10700 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | class |
| **Symbol** | Serializable |
| **Scope** | namespace ara::crypto |
| **Syntax** | class Serializable {...}; |
| **Header file** | #include "ara/crypto/common/serializable.h" |
| **Description** | 可序列化的对象接口。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04003] VolatileTrustedContainer

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04003 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10850 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | class |
| **Symbol** | VolatileTrustedContainer |
| **Scope** | namespace ara::crypto |
| **Syntax** | class VolatileTrustedContainer {...}; |
| **Header file** | #include "ara/crypto/common/volatile\_trusted\_container.h" |
| **Description** | 易失性可信容器的这个显式接口用于缓冲 RAM 中的 CryptoAPI 对象。 此类表示“智能缓冲区”，因为它提供对 IOInterface 的访问，可用于查询缓冲区内容的元数据。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04004] CryptoException

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04004 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_19905 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | class |
| **Symbol** | CryptoException |
| **Base class** | ara::core::Exception |
| **Scope** | namespace ara::crypto |
| **Syntax** | class CryptoException : public Exception {...}; |
| **Header file** | #include "ara/crypto/common/crypto\_error\_domain.h" |
| **Description** | CRYPTO 错误引发的异常类型。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04005] CryptoErrorDomain

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04005 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_19900 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | class |
| **Symbol** | CryptoErrorDomain |
| **Base class** | ara::core::ErrorDomain |
| **Scope** | namespace ara::crypto |
| **Syntax** | class CryptoErrorDomain final : public ErrorDomain {...}; |
| **Unique ID** | 0x8000’0000’0000’0801 |
| **Header file** | #include "ara/crypto/common/crypto\_error\_domain.h" |
| **Description** | 加密错误域类，提供由 ara::core::ErrorDomain 定义的接口，例如加密错误域的名称或每个错误代码的消息。 此类表示一个错误域，负责 ara::crypto 命名空间中的公共 API 可能报告的所有错误。 |
| **Additional** |  |

#### [SWRD-API-Crypto-04006] MakeErrorCode

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04006 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_19951 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | MakeErrorCode(CryptoErrorDomain::Errc code, ara::core::ErrorDomain::SupportDataType data) | |
| **Scope** | namespace ara::crypto | |
| **Syntax** | constexpr ara::core::ErrorCode MakeErrorCode (CryptoErrorDomain::Errc code, ara::core::ErrorDomain::SupportDataType data) noexcept; | |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Parameters (in)** | code | CryptoErrc 枚举中的错误代码标识符 |
| data | 错误描述的补充数据 |
| **Return value** | ara::core::ErrorCode | 根据参数创建的 ErrorCode 实例 |
| **Header file** | #include "ara/crypto/common/crypto\_error\_domain.h" | |
| **Description** | 从加密错误域生成错误代码实例。 返回的 ErrorCode 实例始终引用 CryptoErrorDomain。 | |

#### [SWRD-API-Crypto-04007] LoadCryptoProvider

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04007 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_20099 | |
| **CR** | - | |
| **Consistency** | Yes，Yes:与AutoSar标准一致；No:非标 | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | LoadCryptoProvider(const ara::core::InstanceSpecifier &iSpecify) | |
| **Scope** | namespace ara::crypto | |
| **Syntax** | cryp::CryptoProvider::Uptr LoadCryptoProvider (const ara::core::InstanceSpecifier &iSpecify) noexcept; | |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Parameters (in)** | iSpecify | 需要的CryptoProvider全局唯一标识符 |
| **Return value** | ara::crypto::cryp::CryptoProvider::Uptr | 加载的Crypto Provider唯一智能指针 |
| **Header file** | #include "ara/crypto/common/entry\_point.h" | |
| **Description** | 创建或返回特定 Crypto Provider 的现有单个实例的工厂。 如果 (providerUid== nullptr) 则应加载平台默认提供程序。 | |

#### [SWRD-API-Crypto-04008] LoadKeyStorageProvider

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04008 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30099 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | LoadKeyStorageProvider() | |
| **Scope** | namespace ara::crypto | |
| **Syntax** | keys::KeyStorageProvider::Uptr LoadKeyStorageProvider () noexcept; | |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kRuntimeFault | 如果无法创建 Key Storage Provider 实例 |
| **Return value** | ara::crypto::keys::KeyStorage Provider::Uptr | 指向Key Storage Provider的唯一智能指针 |
| **Header file** | #include "ara/crypto/common/entry\_point.h" | |
| **Description** | 创建或返回密钥存储提供程序的现有单个实例的工厂。 | |

#### [SWRD-API-Crypto-04009] LoadX509Provider

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04009 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_40099 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | LoadX509Provider() | |
| **Scope** | namespace ara::crypto | |
| **Syntax** | x509::X509Provider::Uptr LoadX509Provider () noexcept; | |
| **Exception Safety** | noexcept | |
| **Thread Safety** | Thread-safe | |
| **Errors** | CryptoErrorDomain::kRuntimeFault | 如果无法创建 X.509 Provider 实例 |
| **Return value** | ara::crypto::x509::X509Provider::Uptr | 指向X.509 Provider唯一智能指针 |
| **Header file** | #include "ara/crypto/common/entry\_point.h" | |
| **Description** | 创建或返回 X.509 提供程序的现有单个实例的工厂。 X.509 Provider 应该使用默认的 Crypto Provider 进行散列和签名验证！ 因此，当您加载 X.509 Provider 时，它也会在后台加载默认的 Crypto Provider。 | |

#### [SWRD-API-Crypto-04010 GenerateRandomData

|  |  |  |  |
| --- | --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04010 | | |
| **Type** | Valid | | |
| **Priority** | H | | |
| **Upstream ID** | SWS\_CRYPT\_30098 | | |
| **CR** | - | | |
| **Consistency** | Yes | | |
| **Change Type** | 新增 | | |
| ***Kind*** | function | | |
| **Symbol** | GenerateRandomData(std::uint32\_t count) | | |
| **Scope** | namespace ara::crypto | | |
| **Syntax** | ara::core::Result<ara::core::Vector > GenerateRandom Data (std::uint32\_t count) noexcept; | | |
| **Parameters (in)** | count | | 要生成的随机字节数 |
| **Return value** | ara::core::Result< ara::core::Vector<ara::core::Byte > > | | 填充了生成的随机序列的缓冲区 |
| **Exception Safety** | noexcept | | |
| **Thread Safety** | Thread-safe | | |
| **Errors** | CryptoErrorDomain::kBusyResource | 如果使用的 RNG 当前是无熵的，因此无法提供请求的随机字节数 | |
|  |  |  | |
| **Header file** | #include "ara/crypto/common/entry\_point.h" | | |
| **Description** | 返回一个分配的缓冲区，其中包含所请求大小的生成随机序列。 | | |

#### [SWRD-API-Crypto-04011] ~IOInterface()

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04011 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10810 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | class |
| **Symbol** | ~IOInterface() |
| **Scope** | class ara::crypto::IOInterface |
| **Syntax** | virtual ~IOInterface () noexcept=default; |
| **Header file** | #include "ara/crypto/common/io\_interface.h" |
| **Description** | 析构函数。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04012] GetCapacity

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04012 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10813 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | class | |
| **Symbol** | GetCapacity() | |
| **Scope** | class ara::crypto::IOInterface | |
| **Syntax** | virtual std::size\_t GetCapacity () const noexcept=0; | |
| **Return value** | std::size\_t | 此 IOInterface 的底层缓冲区的容量（以字节为单位） |
| **Header file** | #include "ara/crypto/common/io\_interface.h" | |
| **Description** | 返回底层（槽号）资源的能力。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04013] GetCryptoObjectType

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04013 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10812 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | class | |
| **Symbol** | GetCryptoObjectType() | |
| **Scope** | class ara::crypto::IOInterface | |
| **Syntax** | virtual CryptoObjectType GetCryptoObjectType () const noexcept=0; | |
| **Return value** | CryptoObjectType | 存储在引用资源中的 CryptoObjectType. |
| **Header file** | #include "ara/crypto/common/io\_interface.h" | |
| **Description** | 返回此 IOInterface 引用的对象的 CryptoObjectType。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04014] GetObjectId

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04014 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10811 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | class | |
| **Symbol** | GetObjectId() | |
| **Scope** | class ara::crypto::IOInterface | |
| **Syntax** | virtual CryptoObjectUid GetObjectId () const noexcept=0; | |
| **Return value** | CryptoObjectType | 存储在引用资源中的 CryptoObjectType. |
| **Header file** | #include "ara/crypto/common/io\_interface.h" | |
| **Description** | 返回存储到此 IOInterface 的对象的 COUID。 如果容器为空，则此方法返回 CryptoObjectType::KUndefined。 加密对象的明确标识需要两个组件：CryptoObjectUid 和 CryptoObjectType。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04015] GetPayloadSize

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04015 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10817 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | class | |
| **Symbol** | GetPayloadSize() | |
| **Scope** | class ara::crypto::IOInterface | |
| **Syntax** | virtual std::size\_t GetPayloadSize () const noexcept=0; | |
| **Return value** | std::size\_t | 存储在此 IOInterface 的底层缓冲区中的对象有效负载的大小（以字节为单位） |
| **Header file** | #include "ara/crypto/common/io\_interface.h" | |
| **Description** | 返回存储在此 IOInterface 的底层缓冲区中的对象有效负载的大小。 如果容器为空，则此方法返回 0。返回值不考虑对象的元信息属性，但它们的大小是固定的，并且对于所有加密对象都是通用的，与它们的实际类型无关。 根据对象的实现细节，自动为对象的元信息提供空间。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04016] GetPrimitiveId

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04016 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10822 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | class | |
| **Symbol** | GetPrimitiveId() | |
| **Scope** | class ara::crypto::IOInterface | |
| **Syntax** | virtual CryptoAlgId GetPrimitiveId () const noexcept=0; | |
| **Return value** | CryptoAlgId | the binary Crypto Primitive ID  二进制 Crypto Primitive ID |
| **Header file** | #include "ara/crypto/common/io\_interface.h" | |
| **Description** | Get vendor specific ID of the primitive.  获取供应商指定的算法ID。 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-04017] GetTypeRestriction

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04017 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10818 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | class | |
| **Symbol** | GetTypeRestriction() | |
| **Scope** | class ara::crypto::IOInterface | |
| **Syntax** | virtual CryptoObjectType GetTypeRestriction () const noexcept=0; | |
| **Return value** | CryptoObjectType | 允许内容的对象类型（CryptoObject  Type::kUndefined 表示没有限制） |
| **Header file** | #include "ara/crypto/common/io\_interface.h" | |
| **Description** | 返回此 IOInterface 的内容类型限制。 如果 KeySlotPrototypeProps::mAllowContent TypeChange==TRUE，则应返回 kUndefined。 如果容器具有不同于 CryptoObjectType::kUndefined 的类型限制，则只有提到的类型的对象可以保存到该容器中。 易失性容器没有任何内容类型限制。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04018] IsObjectExportable

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04018 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10816 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | class | |
| **Symbol** | IsObjectExportable() | |
| **Scope** | class ara::crypto::IOInterface | |
| **Syntax** | virtual bool IsObjectExportable () const noexcept=0; | |
| **Return value** | bool | 允许内容的对象类型（CryptoObject  Type::kUndefined 表示没有限制） |
| **Header file** | #include "ara/crypto/common/io\_interface.h" | |
| **Description** | 如果存储到容器的对象设置了“可导出”属性，则为 true。 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-04019] IsVolatile

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04019 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10814 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | class | |
| **Symbol** | IsVolatile() | |
| **Scope** | class ara::crypto::IOInterface | |
| **Syntax** | virtual bool IsVolatile () const noexcept=0; | |
| **Return value** | bool | 如果容器具有易失性（即“临时”或“在 RAM 中”）则为 true，否则为 false |
| **Header file** | #include "ara/crypto/common/io\_interface.h" | |
| **Description** | 返回此 IOInterface 底层缓冲区的易失性。 “会话”对象只能存储到“易失性”容器中。 “易失性”容器的内容将与接口实例一起被销毁。 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-04020 IsValid

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04020 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10823 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | class | |
| **Symbol** | IsValid() | |
| **Scope** | class ara::crypto::IOInterface | |
| **Syntax** | virtual bool IsValid () const noexcept=0; | |
| **Return value** | bool | 如果基础资源（槽号）有效，则为 true，否则为 false |
| **Header file** | #include "ara/crypto/common/io\_interface.h" | |
| **Description** | 获取底层 KeySlot 是否有效。 如果在打开 IOInterface 后修改了基础资源，则 IOInterface 无效。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04021] IsWritable

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04021 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10821 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | class | |
| **Symbol** | IsWritable() | |
| **Scope** | class ara::crypto::IOInterface | |
| **Syntax** | virtual bool IsWritable () const noexcept=0; | |
| **Return value** | bool | 如果可以写入底层资源，则为 true |
| **Header file** | #include "ara/crypto/common/io\_interface.h" | |
| **Description** | 获取底层 KeySlot 是否可写 - 如果此 IOInterface 链接到 VolatileTrusted Container，则始终返回 true。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04022] operator=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04022 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30202 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | class | |
| **Symbol** | operator=(const IOInterface &other) | |
| **Scope** | class ara::crypto::IOInterface | |
| **Syntax** | IOInterface& operator= (const IOInterface &other)=default; | |
| **Parameters (in)** | other | 另一个 instance |
| **Return value** | IOInterface & | \*this, containing the contents of other \*this，包含其他内容 |
| **Header file** | #include "ara/crypto/common/io\_interface.h" | |
| **Description** | 将另一个 IOInterface 复制分配给此实例。 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-04023] operator=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04023 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30203 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | class | |
| **Symbol** | operator=(IOInterface &&other) | |
| **Scope** | class ara::crypto::IOInterface | |
| **Syntax** | IOInterface& operator= (IOInterface &&other)=default; | |
| **Parameters (in)** | other | 另一个 instance |
| **Return value** | IOInterface & | \*this, containing the contents of other \*this，包含其他内容 |
| **Header file** | #include "ara/crypto/common/io\_interface.h" | |
| **Description** | 将另一个 IOInterface 移动分配给该实例。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04024] operator==

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04024 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10150 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator==(const CryptoObjectUid &lhs, const CryptoObjectUid &rhs) | |
| **Scope** | namespace ara::crypto | |
| **Syntax** | constexpr bool operator== (const CryptoObjectUid &lhs, const Crypto ObjectUid &rhs) noexcept; | |
| **Parameters (in)** | lhs | 左侧操作数 |
| rhs | 右侧操作数 |
| **Return value** | bool | 如果所有成员的 lhs 值都等于 rhs，则为 true，否则为 false |
| **Header file** | #include "ara/crypto/common/crypto\_object\_uid.h" | |
| **Description** | CryptoObjectUid 操作数的比较运算符“相等”。 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-04025] operator<

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04025 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10151 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator<(const CryptoObjectUid &lhs, const CryptoObjectUid &rhs) | |
| **Scope** | namespace ara::crypto | |
| **Syntax** | constexpr bool operator< (const CryptoObjectUid &lhs, const Crypto ObjectUid &rhs) noexcept; | |
| **Parameters (in)** | lhs | 左侧操作数 |
| rhs | 右侧操作数 |
| **Return value** | bool | 如果所有成员的 lhs 值都小于 rhs，则为 true，否则为 false |
| **Header file** | #include "ara/crypto/common/crypto\_object\_uid.h" | |
| **Description** | CryptoObjectUid 操作数的比较运算符“小于”。 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-04026] operator>

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04026 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10152 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Operator>(const CryptoObjectUid &lhs, const CryptoObjectUid &rhs) | |
| **Scope** | namespace ara::crypto | |
| **Syntax** | constexpr bool operator> (const CryptoObjectUid &lhs, const Crypto ObjectUid &rhs) noexcept; | |
| **Parameters (in)** | lhs | 左侧操作数 |
| rhs | 右侧操作数 |
| **Return value** | bool | 如果所有成员的 lhs 值都大于 rhs，则为 true，否则为 false |
| **Header file** | #include "ara/crypto/common/crypto\_object\_uid.h" | |
| **Description** | CryptoObjectUid 操作数的比较运算符“大于”。 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-04027] operator!=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04027 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10153 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Operator!=(const CryptoObjectUid &lhs, const CryptoObjectUid &rhs) | |
| **Scope** | namespace ara::crypto | |
| **Syntax** | constexpr bool operator!= (const CryptoObjectUid &lhs, const Crypto ObjectUid &rhs) noexcept; | |
| **Parameters (in)** | lhs | 左侧操作数 |
| rhs | 右侧操作数 |
| **Return value** | bool | 如果 lhs 值中的至少一个成员不等于对应的 rhs的成员，则为 true，否则为 false |
| **Header file** | #include "ara/crypto/common/crypto\_object\_uid.h" | |
| **Description** | CryptoObjectUid 操作数的比较运算符“不等于”。 | |
| **Additional** |  | |

#### [SWRD-API-Crypto-04028] operator<=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04028 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10154 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Operator<=(const CryptoObjectUid &lhs, const CryptoObjectUid &rhs) | |
| **Scope** | namespace ara::crypto | |
| **Syntax** | constexpr bool operator<= (const CryptoObjectUid &lhs, const Crypto ObjectUid &rhs) noexcept; | |
| **Parameters (in)** | lhs | 左侧操作数 |
| rhs | 右侧操作数 |
| **Return value** | bool | 如果lhs的二进制表示小于或等于rhs，则为True，否则为false |
| **Header file** | #include "ara/crypto/common/crypto\_object\_uid.h" | |
| **Description** | CryptoObjectUid 操作数的比较运算符“小于或等于”。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04029] operator>=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04029 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10155 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Operator>=(const CryptoObjectUid &lhs, const CryptoObjectUid &rhs) | |
| **Scope** | namespace ara::crypto | |
| **Syntax** | constexpr bool operator>= (const CryptoObjectUid &lhs, const Crypto ObjectUid &rhs) noexcept; | |
| **Parameters (in)** | lhs | 左侧操作数 |
| rhs | 右侧操作数 |
| **Return value** | bool | 如果lhs的二进制表示大于或等于rhs，则为True，否则为false |
| **Header file** | #include "ara/crypto/common/crypto\_object\_uid.h" | |
| **Description** | CryptoObjectUid 操作数的比较运算符“大于或等于”。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04030 operator==

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04030 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10451 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator==(const Uuid &lhs, const Uuid &rhs) | |
| **Scope** | namespace ara::crypto | |
| **Syntax** | constexpr bool operator== (const Uuid &lhs, const Uuid &rhs) noexcept; | |
| **Parameters (in)** | lhs | 左侧操作数 |
| rhs | 右侧操作数 |
| **Return value** | bool | 如果lhs的二进制表示等于rhs，则为True，否则为false |
| **Header file** | #include "ara/crypto/common/uuid.h" | |
| **Description** | UUid 操作数的比较运算符“等于”。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04031] operator<

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04031 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10452 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator<(const Uuid &lhs, const Uuid &rhs) | |
| **Scope** | namespace ara::crypto | |
| **Syntax** | constexpr bool operator< (const Uuid &lhs, const Uuid &rhs) noexcept; | |
| **Parameters (in)** | lhs | 左侧操作数 |
| rhs | 右侧操作数 |
| **Return value** | bool | 如果lhs的二进制表示小于rhs，则为True，否则为false |
| **Header file** | #include "ara/crypto/common/uuid.h" | |
| **Description** | UUid 操作数的比较运算符“小于”。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04032] operator>

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04032 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10453 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator>(const Uuid &lhs, const Uuid &rhs) | |
| **Scope** | namespace ara::crypto | |
| **Syntax** | constexpr bool operator> (const Uuid &lhs, const Uuid &rhs) noexcept; | |
| **Parameters (in)** | lhs | 左侧操作数 |
| rhs | 右侧操作数 |
| **Return value** | bool | 如果lhs的二进制表示大于rhs，则为True，否则为false |
| **Header file** | #include "ara/crypto/common/uuid.h" | |
| **Description** | UUid 操作数的比较运算符“大于”。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04033] operator!=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04033 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10454 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator!=(const Uuid &lhs, const Uuid &rhs) | |
| **Scope** | namespace ara::crypto | |
| **Syntax** | constexpr bool operator!= (const Uuid &lhs, const Uuid &rhs) noexcept; | |
| **Parameters (in)** | lhs | 左侧操作数 |
| rhs | 右侧操作数 |
| **Return value** | bool | 如果lhs的二进制表示不等于rhs，则为True，否则为false |
| **Header file** | #include "ara/crypto/common/uuid.h" | |
| **Description** | UUid 操作数的比较运算符“不等于”。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04034] operator<=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04034 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10455 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Operator<=(const Uuid &lhs, const Uuid &rhs) | |
| **Scope** | namespace ara::crypto | |
| **Syntax** | constexpr bool operator<= (const Uuid &lhs, const Uuid &rhs) noexcept; | |
| **Parameters (in)** | lhs | 左侧操作数 |
| rhs | 右侧操作数 |
| **Return value** | bool | 如果lhs的二进制表示小于或等于rhs，则为True，否则为false |
| **Header file** | #include "ara/crypto/common/uuid.h" | |
| **Description** | UUid 操作数的比较运算符“小于或等于”。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04035] operator>=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04035 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10456 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Operator>=(const Uuid &lhs, const Uuid &rhs) | |
| **Scope** | namespace ara::crypto | |
| **Syntax** | constexpr bool operator>= (const Uuid &lhs, const Uuid &rhs) noexcept; | |
| **Parameters (in)** | lhs | 左侧操作数 |
| rhs | 右侧操作数 |
| **Return value** | bool | 如果lhs的二进制表示大于或等于rhs，则为True，否则为false |
| **Header file** | #include "ara/crypto/common/uuid.h" | |
| **Description** | UUid 操作数的比较运算符“大于或等于”。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04036] ThrowAsException

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04036 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_19954 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ThrowAsException(const ara::core::ErrorCode &errorCode) | |
| **Scope** | class ara::crypto::CryptoErrorDomain | |
| **Syntax** | void ThrowAsException (const ara::core::ErrorCode &errorCode) const override; | |
| **Parameters (in)** | errorCode | 来自CrytoErrc枚举值中的错误标识符 |
| **Return value** | None |  |
| **Header file** | #include "ara/crypto/common/crypto\_error\_domain.h" | |
| **Description** | 抛出错误代码异常 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04037] CryptoErrorDomain

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04037 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_19954 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | function |
| **Symbol** | CryptoErrorDomain() |
| **Scope** | class ara::crypto::CryptoErrorDomain |
| **Syntax** | constexpr CryptoErrorDomain () noexcept; |
| **Exception Safety:** | noexcept |
| **Header file** | #include "ara/crypto/common/crypto\_error\_domain.h" |
| **Description** | CryptoErrorDomain 的ctor。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04037] Name

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04037 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_19950 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Name() | |
| **Scope** | class ara::crypto::CryptoErrorDomain | |
| **Syntax** | const char\* Name () const noexcept override; | |
| **Return value:** | const char \* | “Crypto”的文本 |
| **Exception Safety:** | noexcept | |
| **Header file** | #include "ara/crypto/common/crypto\_error\_domain.h" | |
| **Description** | 返回“Crypto”的文本 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04038] Message

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04038 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_19953 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | Message(ara::core::ErrorDomain::CodeType errorCode) | |
| **Scope** | class ara::crypto::CryptoErrorDomain | |
| **Syntax** | const char\* Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override； | |
| **Parameters (in):** | errorCode | 来自于CryptoErrc枚举值中的一个错误码 |
| **Return value:** | const char \* | 错误码消息文本 |
| **Exception Safety:** | noexcept | |
| **Header file** | #include "ara/crypto/common/crypto\_error\_domain.h" | |
| **Description** | 将错误代码值转换为文本消息。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04039] ~Serializable()

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04039 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10710 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | function |
| **Symbol** | ~Serializable() |
| **Scope** | class ara::crypto::Serializable |
| **Syntax** | virtual ~Serializable () noexcept=default |
| **Exception Safety:** | noexcept |
| **Header file** | #include "ara/crypto/common/serializable.h" |
| **Description** | 析构函数 |
| **Additional** | - |

#### [SWRD-API-Crypto-04040 ExportPublicly

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04040 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10711 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ExportPublicly(FormatId formatId=kFormatDefault) | |
| **Scope** | class ara::crypto::Serializable | |
| **Syntax** | virtual ara::core::Result<ara::core::Vector<ara::core::Byte> > Export Publicly (FormatId formatId=kFormatDefault) const noexcept=0; | |
| **Parameters (in):** | formatId | Crypto Provider指定的输出数据格式标识符 |
| **Return value:** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 序列化对象数据缓存 |
| **Exception Safety:** | noexcept | |
| **Thread Safety:** | Thread-safe | |
| **Errors:** | CryptoErrorDomain::kInsufficient Capacit | 如果 (output.empty() == false), 但是它的容量小于需要的； |
| CryptoErrorDomain::kUnknown Identifie | 指定的标识符ID未知 |
| CryptoErrorDomain::kUnsupported Forma | 如果指定的标识符是不支持的； |
| **Header file** | #include "ara/crypto/common/serializable.h" | |
| **Description** | 公开序列化自身。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04041] ExportPublicly

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04041 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10712 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | ExportPublicly(FormatId formatId=kFormatDefault) | |
| **Scope** | class ara::crypto::Serializable | |
| **Syntax** | template <typename Alloc = <implementation-defined>> ara::core::Result<ByteVector<Alloc> > ExportPublicly (FormatId format Id=kFormatDefault) const noexcept; | |
| **Template param:** | Alloc | 输出容器的自定义分配器类型 |
| **Parameters (in):** | formatId | Crypto Provider指定的输出数据格式标识符 |
| **Return value:** | ara::core::Result< ara::core::Vector< ara::core::Byte > > | 序列化对象数据缓存 |
| **Exception Safety:** | noexcept | |
| **Thread Safety:** | Thread-safe | |
| **Errors:** | CryptoErrorDomain::kInsufficient Capacit | 如果 (output.empty() == false), 但是它的容量小于需要的； |
| CryptoErrorDomain::kUnknown Identifie | 指定的标识符ID未知 |
| CryptoErrorDomain::kUnsupported Forma | 如果指定的标识符是不支持的； |
| **Header file** | #include "ara/crypto/common/serializable.h" | |
| **Description** | 公开序列化自己。 此方法根据实际保存的值设置输出容器的大小！ | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04042] operator=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04042 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30204 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | class | |
| **Symbol** | operator=(const Serializable &other) | |
| **Scope** | class ara::crypto::Serializable | |
| **Syntax** | Serializable& operator= (const Serializable &other)=default; | |
| **Parameters (in)** | other | 另一个 instance |
| **Return value** | Serializable & | \*this, containing the contents of other \*this，包含其他内容 |
| **Header file** | #include "ara/crypto/common/io\_interface.h" | |
| **Description** | 将另一个Serializable 拷贝分配给该实例。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04043] operator=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04043 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30205 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | class | |
| **Symbol** | operator=(const Serializable &other) | |
| **Scope** | class ara::crypto::Serializable | |
| **Syntax** | Serializable& operator= (const Serializable &&other)=default; | |
| **Parameters (in)** | other | 另一个 instance |
| **Return value** | Serializable & | \*this, containing the contents of other \*this，包含其他内容 |
| **Header file** | #include "ara/crypto/common/io\_interface.h" | |
| **Description** | 将另一个Serializable 移动分配给该实例。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04044] ~VolatileTrustedContainer

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04044 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10851 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | function |
| **Symbol** | ~VolatileTrustedContainer() |
| **Scope** | class ara::crypto::VolatileTrustedContainer |
| **Syntax** | virtual ~VolatileTrustedContainer () noexcept=default |
| **Exception Safety:** | noexcept |
| **Header file** | #include "ara/crypto/common/volatile\_trusted\_container.h" |
| **Description** | 析构函数 |
| **Additional** | - |

#### [SWRD-API-Crypto-04044] ~VolatileTrustedContainer

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04044 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10851 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | function |
| **Symbol** | ~VolatileTrustedContainer() |
| **Scope** | class ara::crypto::VolatileTrustedContainer |
| **Syntax** | virtual ~VolatileTrustedContainer () noexcept=default |
| **Exception Safety:** | noexcept |
| **Header file** | #include "ara/crypto/common/volatile\_trusted\_container.h" |
| **Description** | 析构函数 |
| **Additional** | - |

#### [SWRD-API-Crypto-04045] GetIOInterface

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04045 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10853 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | GetIOInterface() | |
| **Scope** | class ara::crypto::VolatileTrustedContainer | |
| **Syntax** | virtual IOInterface& GetIOInterface () const noexcept=0; | |
| **Return value:** | IOInterface & | 此容器的 IOInterface的引用 |
| **Exception Safety:** | noexcept | |
| **Header file** | #include "ara/crypto/common/volatile\_trusted\_container.h" | |
| **Description** | 检索用于将对象导入/导出到此容器的 IOInterface | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04046] operator=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04046 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30206 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator=(const VolatileTrustedContainer &other) | |
| **Scope** | class ara::crypto::VolatileTrustedContainer | |
| **Syntax** | VolatileTrustedContainer& operator= (const VolatileTrustedContainer &other)=default; | |
| **Parameters (in):** | other | 另一个实例 |
| **Return value:** | VolatileTrustedContainer & | \*this, containing the contents of other |
| **Exception Safety:** | noexcept | |
| **Header file** | #include "ara/crypto/common/volatile\_trusted\_container.h" | |
| **Description** | 将另一个 VolatileTrustedContainer 复制分配给此实例。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04047] operator=

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04047 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_30206 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | operator=(const VolatileTrustedContainer &&other) | |
| **Scope** | class ara::crypto::VolatileTrustedContainer | |
| **Syntax** | VolatileTrustedContainer& operator= (const VolatileTrustedContainer &&other)=default; | |
| **Parameters (in):** | other | 另一个实例 |
| **Return value:** | VolatileTrustedContainer & | \*this, containing the contents of other |
| **Exception Safety:** | noexcept | |
| **Header file** | #include "ara/crypto/common/volatile\_trusted\_container.h" | |
| **Description** | 将另一个 VolatileTrustedContainer 移动分配给此实例。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04048] IsNil

|  |  |  |
| --- | --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04048 | |
| **Type** | Valid | |
| **Priority** | H | |
| **Upstream ID** | SWS\_CRYPT\_10411 | |
| **CR** | - | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind*** | function | |
| **Symbol** | IsNil() | |
| **Scope** | struct ara::crypto::Uuid | |
| **Syntax** | bool IsNil () const noexcept; | |
| **Return value:** | bool | 如果这个标识符是“Nil”,则为true;否则为false; |
| **Exception Safety:** | noexcept | |
| **Thread Safety:** | Thread-safe | |
| **Header file** | #include "ara/crypto/common/uuid.h" | |
| **Description** | 检查此标识符是否为“Nil UUID”（根据 RFC4122）。 | |
| **Additional** | - | |

#### [SWRD-API-Crypto-04049] kAlgIdUndefined

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04049 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13000 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAlgIdUndefined |
| **Scope** | namespace ara::crypto |
| **Type:** | const CryptoAlgId |
| **Syntax:** | const CryptoAlgId kAlgIdUndefined = 0u; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 算法 ID 未定义。 此值也可用于以下含义：任何或默认算法，无算法。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04050 kAlgIdAny

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04050 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13001 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAlgIdAny |
| **Scope** | namespace ara::crypto |
| **Type:** | const CryptoAlgId |
| **Syntax:** | const CryptoAlgId kAlgIdAny = kAlgIdUndefined; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 允许任何算法 ID。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04051] kAlgIdDefault

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04051 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13002 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAlgIdDefault |
| **Scope** | namespace ara::crypto |
| **Type:** | const CryptoAlgId |
| **Syntax:** | const CryptoAlgId kAlgIdDefault = kAlgIdUndefined; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 默认算法 ID（在当前上下文/原语中）。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04052] kAlgIdNone

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04052 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13003 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAlgIdNone |
| **Scope** | namespace ara::crypto |
| **Type:** | const CryptoAlgId |
| **Syntax:** | const CryptoAlgId kAlgIdNone = kAlgIdUndefined; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 没有算法 ID（即算法定义不适用）。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04053] kAllowDataDecryption

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04053 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13102 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAlgIdUndefined |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowDataDecryption = 0x0002; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 密钥/种子可用于数据解密初始化（适用于对称和非对称算法）。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04054] kAllowDataEncryption

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04054 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13101 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAlgIdUndefined |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowDataEncryption = 0x0001; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 密钥/种子可用于数据加密初始化（适用于对称和非对称算法）。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04055] kAllowDerivedDataDecryption

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04055 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13113 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowDerivedDataDecryption |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowDerivedDataDecryption = kAllowData Decryption << 16; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 派生的种子或对称密钥可用于数据解密。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04056] kAllowDerivedDataEncryption

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04056 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13112 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAlgIdUndefined |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowDerivedDataEncryption = kAllowData Encryption << 16; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 派生的种子或对称密钥可用于数据加密。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04057] kAllowDerivedRngInit

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04057 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13117 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowDerivedRngInit |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowDerivedRngInit = kAllowRngInit << 16; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 派生的种子或对称密钥可用于为 RandomGeneratorContext 播种。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04058] kAllowDerivedExactModeOnly

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04058 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13121 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowDerivedExactModeOnly |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowDerivedExactModeOnly = kAllowExactMode Only << 16; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 仅将派生对象的使用限制为指定的操作模式。 派生的种子或对称密钥只能用于 Key::AlgId 直接指定的模式。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04059] kAllowDerivedKdfMaterial

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04059 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13118 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowDerivedKdfMaterial |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowDerivedKdfMaterial = kAllowKdfMaterial << 16; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 派生的种子或对称密钥可用作 RestrictedUseObject，用于通过密钥派生函数 (KDF) 派生从属密钥。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04060 kAllowKdfMaterialAnyUsage

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04060 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13122 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowKdfMaterialAnyUsage |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowKdfMaterialAnyUsage = kAllowKdfMaterial | kAllowDerivedDataEncryption | kAllowDerivedDataDecryption | kAllow DerivedSignature | kAllowDerivedVerification | kAllowDerivedKey Diversify | kAllowDerivedRngInit | kAllowDerivedKdfMaterial | kAllow DerivedKeyExporting | kAllowDerivedKeyImporting |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 允许将对象用作 KDF 的关键材料以及派生对象的任何使用。 种子或对称密钥可以用作密钥派生函数 (KDF) 的 RestrictedUseObject，并且派生的“从属”密钥可以不受限制地使用。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04061] kAllowDerivedKeyDiversify

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04061 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13116 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowDerivedKeyDiversify |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowDerivedKeyDiversify = kAllowKeyDiversify << 16; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 派生的种子或对称密钥可用于从属密钥多样化。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04062] kAllowDerivedKeyExporting

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04062 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13119 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowDerivedKeyExporting |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowDerivedKeyExporting = kAllowKeyExporting << 16; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 派生的种子或对称密钥可用作密钥包装转换的“传输”密钥 |
| **Additional** | - |

#### [SWRD-API-Crypto-04063] kAllowDerivedKeyImporting

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04063 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13120 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowDerivedKeyImporting |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowDerivedKeyImporting = kAllowKeyImporting << 16; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 派生的种子或对称密钥可用作密钥解包转换的“传输”密钥。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04064] kAllowDerivedSignature

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04064 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13114 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowDerivedSignature |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowDerivedSignature = kAllowSignature << 16; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 派生的种子或对称密钥可用于 MAC/HMAC 生产; |
| **Additional** | - |

#### [SWRD-API-Crypto-04065] kAllowDerivedVerification

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04065 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13115 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowDerivedVerification |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowDerivedVerification = kAllowVerification << 16; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 派生的种子或对称密钥可用于 MAC/HMAC 验证。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04066] kAllowExactModeOnly

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04066 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13111 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowExactModeOnly |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowExactModeOnly = 0x8000; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 密钥只能用于 Key::AlgId 直接指定的模式。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04067] kAllowKdfMaterial

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04067 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13108 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowKdfMaterial |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowKdfMaterial = 0x0080; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 该对象可用作 KDF 的输入密钥材料。 种子或对称密钥可以用作 RestrictedUseObject，用于通过密钥派生函数 (KDF) 派生从属密钥。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04068] kAllowKeyAgreement

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04068 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13105 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowKeyAgreement |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowKeyAgreement = 0x0010; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 种子或非对称密钥可用于密钥协商协议的执行。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04069] kAllowKeyDiversify

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04069 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13106 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowKeyDiversify |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowKeyDiversify = 0x0020; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 种子或对称密钥可用于从属密钥多样化。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04070 kAllowKeyExporting

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04070 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13109 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowKeyExporting |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowKeyExporting = 0x0100; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 种子或对称密钥可用于从属密钥多样化。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04071] kAllowKeyImporting

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04071 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13110 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowKeyImporting |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowKeyImporting = 0x0200 |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 密钥可用作密钥包装或封装转换的“传输”密钥（适用于对称和非对称密钥）。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04072] kAllowPrototypedOnly

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04072 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13100 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowPrototypedOnly |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowPrototypedOnly = 0; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 该组包含为允许使用标志预定义的 1 位常量值列表。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04073] kAllowRngInit

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04073 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13107 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowRngInit |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowRngInit = 0x0040; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 种子或对称密钥可用于RandomGeneratorCtx 的种子 |
| **Additional** | - |

#### [SWRD-API-Crypto-04074] kAllowSignature

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04074 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13103 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowSignature |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowSignature = 0x0004; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 密钥/种子可用于数字签名或 MAC/HMAC 生产（适用于对称和非对称算法）。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04075] kAllowVerification

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04075 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13104 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kAllowVerification |
| **Scope** | namespace ara::crypto |
| **Type:** | const AllowedUsageFlags |
| **Syntax:** | const AllowedUsageFlags kAllowVerification = 0x0008; |
| **Header file** | #include "ara/crypto/common/base\_id\_types.h" |
| **Description** | 密钥/种子可用于数字签名或 MAC/HMAC 验证（适用于对称和非对称算法）。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04076] mVersionStamp

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04076 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_13102 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | mVersionStamp |
| **Scope** | struct ara::crypto::CryptoObjectUid |
| **Type:** | std::uint64\_t |
| **Syntax:** | std::uint64\_t mVersionStamp = 0u; |
| **Header file** | #include "ara/crypto/common/crypto\_object\_uid.h" |
| **Description** | 稳定定时器或简单计数器的序列值，代表对应加密对象的版本。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04077] mLSQW

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04077 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30002 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | mLSQW |
| **Scope** | struct ara::crypto::SecureCounter |
| **Type:** | std::uint64\_t |
| **Syntax:** | std::uint64\_t mLSQW; |
| **Header file** | #include "ara/crypto/common/entry\_point.h" |
| **Description** | 最低有效 64 位 |
| **Additional** | - |

#### [SWRD-API-Crypto-04078] mMSQW

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04078 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_30003 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | mMSQW |
| **Scope** | struct ara::crypto::SecureCounter |
| **Type:** | std::uint64\_t |
| **Syntax:** | std::uint64\_t mMSQW; |
| **Header file** | #include "ara/crypto/common/entry\_point.h" |
| **Description** | 最高有效 64 位 |
| **Additional** | - |

#### [SWRD-API-Crypto-04079] kFormatDefault

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04079 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10750 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kFormatDefault |
| **Scope** | class ara::crypto::Serializable |
| **Type:** | const FormatId |
| **Syntax:** | static const FormatId kFormatDefault = 0; |
| **Header file** | #include "ara/crypto/common/serializable.h" |
| **Description** | 默认序列化格式 |
| **Additional** | - |

#### [SWRD-API-Crypto-04080 kFormatDerEncoded

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04080 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10752 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kFormatDerEncoded |
| **Scope** | class ara::crypto::Serializable |
| **Type:** | const FormatId |
| **Syntax:** | static const FormatId kFormatDerEncoded = 2; |
| **Header file** | #include "ara/crypto/common/serializable.h" |
| **Description** | 导出对象的 DER 编码值。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04081] kFormatPemEncoded

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04081 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10753 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kFormatPemEncoded |
| **Scope** | class ara::crypto::Serializable |
| **Type:** | const FormatId |
| **Syntax:** | static const FormatId kFormatPemEncoded = 3; |
| **Header file** | #include "ara/crypto/common/serializable.h" |
| **Description** | 导出对象的 PEM 编码值。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04082] kFormatRawValueOnly

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04082 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10751 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | kFormatRawValueOnly |
| **Scope** | class ara::crypto::Serializable |
| **Type:** | const FormatId |
| **Syntax:** | static const FormatId kFormatRawValueOnly = 1; |
| **Header file** | #include "ara/crypto/common/serializable.h" |
| **Description** | 仅导出对象的原始值。 |
| **Additional** | - |

#### [SWRD-API-Crypto-04083] mQwordLs

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04083 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10412 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | mQwordLs |
| **Scope** | struct ara::crypto::Uuid |
| **Type:** | std::uint64\_t |
| **Syntax:** | std::uint64\_t mQwordLs = 0u; |
| **Header file** | #include "ara/crypto/common/uuid.h" |
| **Description** | 低位的 QWORD |
| **Additional** | - |

#### [SWRD-API-Crypto-04084] mQwordMs

|  |  |
| --- | --- |
| **SWRD-ID** | SWRD-API-Crypto-04084 |
| **Type** | Valid |
| **Priority** | H |
| **Upstream ID** | SWS\_CRYPT\_10413 |
| **CR** | - |
| **Consistency** | Yes |
| **Change Type** | 新增 |
| ***Kind*** | variable |
| **Symbol** | mQwordMs |
| **Scope** | struct ara::crypto::Uuid |
| **Type:** | std::uint64\_t |
| **Syntax:** | std::uint64\_t mQwordMs = 0u; |
| **Header file** | #include "ara/crypto/common/uuid.h" |
| **Description** | 高位的QWORD |
| **Additional** | - |

附录A- 信息定义

|  |  |  |
| --- | --- | --- |
| 类别 | 结构 | 备注 |
| SWRD-ID | SWRD-{需求类型}-{功能简称}-流水号  功能简称：参见下面功能简称列表  需求类型：功能需求为空，非功能需求为NF,接口为API  流水号：从00001开始的5位自然数 | *例：*  *SWRD-Nvm-00001*  *SWRD-NF\_Nvm-00001*  *SWRD-API-Nvm-00001*  *SWRD-Crypto-00001* |

|  |  |
| --- | --- |
| 功能简称列表(aCore) | 说明 |
| DM\_DEM | Diagnostics management模块的诊断事件管理 |
| DM\_DCM | Diagnostics management模块的诊断通信管理 |
| DM\_DCM\_DOIP | Diagnostics management模块的DO/IP相关功能 |
| CoreTypes | 核心数据类型 |
| CM\_CommunicationAPI | Communication management模块的Communication API相关功能 |
| CM\_SOMEIP | Communication management模块的SOME/IP相关功能 |
| CM\_DDS | Communication management模块的DDS相关功能 |
| CM\_CommunicationGroup | Communication management模块的通信组相关功能 |
| CM\_SHM | Communication management模块的共享内存相关功能 |
| CM\_IPC | Communication management模块的IPC相关功能 |
| CM\_Raw | Communication management模块的raw data streaming相关功能 |
| CM\_TLS | Communication management模块的TLS相关功能 |
| CM\_S2S | Communication management模块的S2S相关功能 |
| CM\_E2E | Communication management模块的E2E相关功能 |
| UCM\_Master | Update and config management模块的主站相关功能 |
| UCM\_Server | Update and config management模块的从站相关功能 |
| LT | Log and trace模块相关功能 |
| PHM | Platform health management模块相关功能 |
| Per | Persistency模块相关功能 |
| SM | State management模块相关功能 |
| Crypto | Cryptography模块相关功能 |
| EM | Execution mangement模块相关功能 |
| NM | Network management模块相关功能 |
| TS | Time synchronization模块相关功能 |

说明：根据项目情况可自己定义，增加功能简称

|  |  |  |
| --- | --- | --- |
| 安全等级(ASIL) | 解释说明 | 备注 |
| ASIL A | 根据S – Severity(严重度)  E – Exposure(暴露度) C – Controllability(可控性) 排定功能安全等级。详细理解可以参考26262标准文件。 | *如果有关于ASIL等级的特殊解释说明，请记录在此* |
| ASIL B |  |
| ASIL C |  |
| ASIL D |  |
| QM(A) | 从ASIL A到ASIL D 中分解出来，分解的标准，参考功能安全体系文件《功能安全需求分解指南\_FS.pdf》  其中:括号内的等级为原始等级。 |  |
| QM(B) |  |
| QM(C) |  |
| QM(D) |  |
| ASIL A(A) |  |
| ASIL A(B) |  |
| ASIL A(C) |  |
| ASIL A(D) |  |
| ASIL B(B) |  |
| ASIL B(C) |  |
| ASIL B(D) |  |
| ASIL C(C) |  |
| ASIL C(D) |  |
| ASIL D(D) |  |

|  |  |  |
| --- | --- | --- |
| 优先级(Priority) | 解释说明 | 备注 |
| H | 高优先级 | *例：被依赖的需求优先级设置为H级别* |
| M | 中优先级 | *例：* |
| L | 低优先级 | *例：其余功能均设置为L级别* |
| N/A | 不适用 |  |

|  |  |  |
| --- | --- | --- |
| 类型  (Type) | 状态说明 | 备注 |
| Valid | 有效 | *例：表示需要对应* |
| InValid | 不适用 | *例：表示不做对应* |
| TBD | 检讨中 | *例：表示正在检讨中* |

说明：根据项目情况可自己定义，但需要明确

|  |  |  |
| --- | --- | --- |
| 状态  (Status) | 状态说明 | 备注 |
| Draft | 草稿 | *例：表示新建* |
| In Review | 评审中 | *例：表示处于评审中* |
| Approved | 批准 | *例：表示通过评审* |
| Released | 发布 | *例：表示通过客户确认* |
| Modified | 修改 | *例：表示正在检讨修改中* |

|  |  |  |
| --- | --- | --- |
| 变更类型  (Change Type) | 解释说明 | 备注 |
| 新增 | 与AutoSAR标准*XXX*相比，新增的需求。 | 如果有关于每个变更类型的特殊解释说明，请记录在此 |
| 修改 | 与AutoSAR标准*XXX*相比，发生了修改的需求 |  |
| 不变 | 与AutoSAR标准*XXX*相比，没有变更的需求。 |  |
| 删除 | 与AutoSAR标准*XXX*相比，没有变更的需求。 |  |

说明：根据项目情况可自己定义，但需要明确

附录B- 配置信息

|  |  |  |  |
| --- | --- | --- | --- |
| 配置信息 | 说明 | 范围 | 备注 |
| *API configuration class* |  | *1、2、3* |  |
|  |  |  |  |
|  |  |  |  |