**镜像签名认证软件研制技术方案**

承研单位 西安电子科技大学

负 责 人 李龙海

联系电话 18992864681

时 间 2025 年 4 月

[1. 范围 1](#_Toc152661158)

[1.1 项目概述 1](#_Toc1680607988)

[1.2 需求分析 3](#_Toc126045325)

[1.2.1 功能需求 4](#_Toc1024901333)

[1.2.2 非功能需求 8](#_Toc550371144)

[2 总体设计 8](#_Toc875749579)

[2.1 运行环境（没写） 8](#_Toc2017741362)

[2.1.1 软件环境 8](#_Toc1264801357)

[2.1.2 硬件环境 9](#_Toc1723269093)

[2.2 架构设计 9](#_Toc2019182609)

[2.3 外部接口 10](#_Toc1865519569)

[2.4 软件部件 13](#_Toc526149983)

[2.5 技术方案 17](#_Toc1812589582)

[2.5.1 容器镜像签名与管理模块方案 17](#_Toc2137571979)

[2.5.2 TUF元数据校验模块 25](#_Toc918320390)

[2.5.3 角色与密钥管理模块方案 29](#_Toc245823741)

[2.5.4 信任仓库与目标文件管理模块 34](#_Toc1948561806)

[2.5.5 通用模块技术方案 37](#_Toc352656692)

[2.6 内部接口设计 42](#_Toc46156724)

[2.6.1 容器镜像签名与管理模块接口 42](#_Toc514463701)

[2.6.2 角色与密钥管理模块接口 44](#_Toc822259885)

[2.6.3 信任仓库与目标文件管理模块接口 45](#_Toc664618750)

[2.6.4 通用模块接口 46](#_Toc1184883203)

[3 测试与验证策略 48](#_Toc716134190)

[3.1 单元测试策略 48](#_Toc1568973542)

[3.2 集成测试策略 49](#_Toc786618881)

[3.3 系统测试策略 50](#_Toc794202035)

[3.3.1 功能测试方法 50](#_Toc1542736140)

[3.3.2 性能测试方法 50](#_Toc48751102)

[4 项目验收与交付 51](#_Toc1168501807)

# 范围

## 项目概述

**（1）项目背景**

在容器化和虚拟化广泛应用的现代计算环境中，软件镜像成为应用部署和分发的核心载体。镜像的可信性和完整性直接关系到系统运行的安全性和稳定性。随着软件供应链攻击事件频发，保障镜像内容来源可验证、签名可信、内容未被篡改，已成为构建安全、可靠基础设施的重要技术需求。

镜像签名认证机制通过对镜像生成数字签名，并在分发与拉取过程中进行验证，可有效防止恶意篡改、伪造镜像等攻击行为。同时，结合软件更新元数据管理机制，可以实现对镜像版本的可追溯、可回滚管理，提升镜像生命周期的可控性与安全性。

The Update Framework（TUF）作为一种安全的软件更新框架，提出了角色划分、元数据签名、密钥轮换等机制，能够有效抵御元数据伪造、密钥泄露等复杂攻击。基于 TUF 的签名认证机制，不仅具有强安全性保障，还具备良好的可扩展性和跨平台适应能力，适用于多种镜像分发场景。

然而，当前实际应用中仍存在若干亟待解决的问题：

a. 缺乏一套独立、可控、适用于本地化部署或私有环境的镜像签名认证系统，难以满足对数据安全性、系统可控性要求较高的业务场景；

b. 现有方案对签名算法和封装格式的支持有限，难以灵活切换或扩展以满足特定算法或标准需求，且密钥轮换策略难以配置，不能满足高安全性场景的密钥生命周期管理需求；

c. 缺少灵活的信任库管理机制与签名元数据服务组件，难以满足大规模多用户环境下的权限控制、签名策略隔离等需求。

因此，研发一套基于 TUF 模型的通用镜像签名认证软件，具有重要意义。该软件将具备客户端与服务端功能模块，支持镜像签名与验证、元数据生成与管理、密钥管理与轮换、安全通信等核心能力，并支持多种签名算法和签名封装格式，适配多类型镜像对象。该系统能够为多样化部署环境下的镜像可信分发提供统一、可控、安全的技术支撑，提升镜像分发体系的整体可信性与安全性。

**（2）使用场景、功能简要介绍**

针对上述问题，本项目旨在针对嵌入式容器云平台研制一套镜像签名认证软件系统，面向容器镜像及虚拟机镜像等镜像分发对象，提供一整套符合 TUF模型设计思想的签名、验证、信任管理能力，适用于私有仓库、企业内部部署环境、安全隔离场景下的镜像可信分发体系构建。

本项目研制的镜像签名认证软件包含客户端与服务端两部分，它们的应用场景和主要功能如图1所示。



图1 应用场景和功能示意图

镜像签名认证软件主要用于容器镜像生命周期中的内容信任控制与签名认证管理，旨在提升容器镜像的安全性与可追溯性。该系统基于TUF（The Update Framework）构建，提供从镜像签名生成、信任验证到密钥管理的全流程能力，支持ECDSA、RSA-PSS等加密算法与JWS签名封装格式。

镜像签名认证软件包含客户端与服务器两部分，客户端工具支持开发者本地初始化信任库、管理密钥、生成镜像签名并将元数据发布至服务器端，客户端工具能够与Docker进行集成并兼容天脉操作系统；服务器端负责集中管理TUF元数据、执行签名校验、冲突检测与密钥轮换等任务，确保签名策略的统一性与数据的一致性。该软件还提供可配置的信任策略开关，可强制容器云平台在推送与拉取镜像时执行签名验证流程，保障仅已签名、经认证的镜像被使用。

对于**容器应用开发人员**，可使用客户端工具将构建完成的容器镜像进行签名，生成对应的信任元数据并上传至签名认证服务器端；对于**部署与运维人员**，系统提供了镜像拉取过程中的内容信任验证功能。

当开启镜像信任策略时，容器云平台客户端仅允许拉取通过签名认证服务验证的可信镜像，确保部署到生产环境中的每一个镜像都是经由开发人员签名、具备可溯源性的安全版本。

## 需求分析

镜像签名认证软件可在本地或远程环境中初始化镜像信任仓库，生成并维护基于 TUF（The Update Framework）标准的元数据，支持自定义元数据存储路径及安全密钥加密存储机制。软件支持自动生成、管理根密钥（Root Key）、目标密钥（Target Key）、时间戳密钥及快照密钥，满足从镜像构建、签名、上传到拉取、验证等各阶段的安全性要求。

镜像签名认证软件需要嵌入现有的容器镜像构建、发布与部署的系统架构中，实现对镜像操作行为的细粒度信任策略控制。支持设定镜像信任验证开关，允许用户强制启用内容信任策略，从而使容器云平台客户端仅可操作通过签名认证的软件镜像。

在复杂的镜像签名与信任策略场景中，软件需要支持灵活的密钥轮换策略及委托角色管理机制。支持基于时间周期、密钥状态或策略变更的自动密钥轮换，满足长期运行环境中的信任维护需求；同时支持委托角色的添加、删除与授权控制，便于组织内部按职责划分签名权限，实现分级安全防护。

镜像签名认证软件需要提供完整的服务端功能模块，包括信任库管理、元数据存储与验证、签名发布与吊销等能力。支持客户端与服务端间的安全通信、算法多样性扩展（支持 ECDSA 与 RSA-PSS）、JWS 格式签名封装、关键数据备份恢复与日志记录等辅助管理功能。

### 功能需求

镜像签名认证软件分为客户端软件与服务器软件。

客户端软件的主要功能包括：（1）信任库与目标文件管理功能，支持本地信任库的初始化与配置，生成符合 TUF标准的元数据文件，并对目标镜像文件进行管理，保障本地签名体系的完整性；（2）密钥与角色管理功能，提供本地密钥的生成、加密存储、列举与轮换功能，支持根密钥与目标密钥的创建与替换以及密钥的管理功能；（3）容器镜像签名与管理功能，提供对容器镜像进行签名、验证和撤销的功能，支持设置镜像信任验证开关，提供镜像元数据签名的发布、信任信息的查询、签名链的验证显示，确保镜像在传输和部署过程中的可信性。（4）辅助管理功能，支持配置签名认证服务器地址、本地密钥安全存储路径，提供客户端帮助信息的查看功能，提升系统的易用性和可维护性。

服务器软件的主要功能包括：（1）信任库与目标文件管理功能，支持服务端信任库的初始化与配置，保证服务端签名状态与客户端保持一致；（2）TUF元数据管理功能，提供元数据的存储、签名、更新以及客户端提交元数据的格式与签名验证，确保签名数据一致性与可信度；（3）密钥与角色管理功能，支持服务端密钥（如时间戳密钥、快照密钥）的生成、列举、签名与吊销操作，并按角色分层实施密钥管理策略，保障签名链的有效性和安全性；（4）辅助管理功能：支持通过配置文件定义服务器行为、安全存储设定及服务端帮助信息查看，增强服务端运行的灵活性和可管理性。

软件还包含一些客户端与服务端通用功能，主要包括：（1）安全通信功能：支持客户端与服务端之间的安全通信通道建立，保障签名与验证过程中的数据传输安全性；（2）加密与签名算法支持功能：支持多种签名算法（如ECDSA、RSA-PSS），具备算法可扩展性，以适应不同安全合规需求。（3）签名封装功能：支持使用 JWS（JSON Web Signature）格式封装签名内容，便于签名的标准化和跨平台验证。（4）数据管理与日志功能。

镜像签名认证软件客户端功能要求如下：

（1）信任（仓）库与目标文件管理相关功能：

1. 提供镜像签名认证软件信任（仓）库初始化功能；

——支持初始化本地信任库，并生成TUF（The Update Framework）相关元数据文件的功能；

——支持自定义配置TUF相关元数据文件存储路径功能；

——支持自动生成根密钥（Root Key）、目标密钥（Target Key）；

——支持密钥加密存储的功能；其中私钥需要加密存储，其密码可通过环境变量或交互方式输入。

1. 提供镜像签名认证软件信任库内容列举功能；
2. 提供镜像签名认证软件信任库对目标文件的增、删、更、查功能；
3. 提供快照元数据生成并签名功能。

（2）角色与密钥管理相关功能：

1. 提供本地存储所有密钥列举功能；
2. 提供新公私钥对生成功能；

——提供根密钥（Root Key）对，私钥需加密存储，公钥发布；

——提供目标密钥（Targets Key）对，私钥需加密存储，公钥发布。

1. 提供密钥轮换功能；

——支持策略变更：当信任策略更新（如新增/移除签名角色）时自动关联密钥轮换；

——提供密钥临期、逾期使用时告警，启动轮换流程；

——定期自动轮换：支持按时间周期自动触发密钥轮换；

——支持时间可配置轮换策略，按角色（Root、Targets、Snapshot）分层轮换；

1. 提供委托角色（密钥）添加、回收、列举功能。

（3）容器镜像签名与管理相关功能：

1. 提供为镜像创建签名功能；
2. 提供镜像信任验证开关设定功能；

——支持开启镜像信任开关时，强制启用镜像签名认证软件内容信任策略，Docker客户端仅允许推送和拉取其已签名的镜像，且必须通过镜像签名认证软件服务验证签名的有效性。

——支持关闭镜像信任开关时，禁用内容信任，Docker 客户端允许操作未签名镜像，绕过镜像签名认证软件的签名验证流程。

1. 提供镜像元数据签名发布功能；
2. 提供查看镜像信任信息功能；

——支持查看目标文件签名元数据的详细信息，包括签名者身份、使用的密钥、签名时间；

——支持显示签名者公钥及其对应的密钥ID；

——支持验证信任链完整性功能；

——支持密钥层级的显示功能。

1. 提供撤销指定镜像信任功能；

——支持元数据修改和版本更新；

——支持向服务器端同步信任状态。

1. 提供密钥吊销检查的功能，发起吊销请求、生成签名元数据更新；
2. 提供推送镜像时自动签名功能，支持信任链构建，完成签名生成与上传；
3. 提供对拉取的镜像信任验证功能，支持信任链和签名的验证。

（4）辅助管理功能：

1. 提供镜像签名认证软件服务器设定功能；
2. 提供本地密钥文件安全存储信息设定功能；
3. 提供客户端使用帮助信息查看功能。

镜像签名认证软件服务器软件功能需求如下：

（1）信任（仓）库与目标文件管理相关功能

1. 提供镜像签名认证软件信任（仓）库初始化功能；
2. 提供镜像签名认证软件信任库内容列举功能；
3. 提供镜像签名认证软件信任库对目标文件的增、删、更、查功能；
4. 提供服务端快照元数据生成并签名功能。

（2）TUF元数据管理相关功能

1. 提供元数据存储与更新的功能；
2. 提供目标文件元数据验证与冲突检测功能，支持对客户端提交的元数据进行格式、签名及版本的验证。

（3）角色与密钥管理相关功能

1. 提供本地存储所有密钥列举功能；
2. 提供新公私钥对（时间戳密钥、快照密钥）生成功能；
3. 提供时间戳和快照元数据签名的功能；
4. 提供密钥吊销检查的功能；
5. 提供密钥轮换功能；

——支持策略变更：当信任策略更新（如新增/移除签名角色）时自动关联密钥轮换；

——提供密钥临期、逾期使用时告警，启动轮换流程；

——定期自动轮换：支持按时间周期自动触发密钥轮换；

——支持可配置轮换策略，按角色（时间戳密钥、快照密钥）分层轮换。

（3）辅助管理功能

1. 提供镜像签名认证软件服务器设定功能：支持通过配置文件定义服务端行为；
2. 提供本地密钥文件安全存储信息设定功能；
3. 提供服务端使用帮助信息查看功能。

镜像签名认证软件客户端与服务端通用的功能：

1. 提供服务端与客户端交互安全通信通道功能；
2. 提供多种算法支持和扩展性的功能；

——支持加密算法兼容性：支持ECDSA、RSA-PSS算法；

——支持签名算法可扩展性，用户可替换使用私有的签名算法的功能。

1. 提供JWS格式的签名封装；
2. 提供关键数据的备份恢复功能；
3. 提供日志记录的功能。

### 非功能需求

#### 性能要求

1. 镜像签名认证软件服务器可支持不少于1000个的软仓库公证维护；
2. 镜像签名认证软件服务器至少可支持100个镜像签名认证软件客户端的并发访问；
3. 时间戳元数据文件生成及签名延时0.5秒；
4. 快照元数据文件生成及签名延时0.5秒；

#### 软件要求

1. 兼容天脉操作系统
2. 镜像签名认证客户端能够与docker完成集成验证
3. 采用C/C++语言开发
4. 编码标准遵循GJB 8114-2013《C/C++语言编程安全子集》。

# 总体设计

## 运行环境（没写）

### 软件环境

镜像签名认证软件系统的软件环境如表1所示。

表1 镜像签名认证软件系统软件环境

|  |  |
| --- | --- |
| 集群环境 |  |
| 目标机环境 |  |
| 宿主机环境 |  |
| 开发语言 | C++ |
| 开发工具 | GCC、CMake |
| 调试工具 | GDB |

### 硬件环境

## 架构设计



图 2 系统架构设计图

签名认证软件的总体架构设计如图 2所示，整体分为客户端与服务端两大部分，包含命令交互工具模块、容器镜像签名与管理模块、TUF元数据校验与存储模块、信任仓库与目标文件管理模块、角色与密钥管理模块、通用模块等若干子系统，各模块共同协作实现镜像的可信签名与认证功能，保障镜像在发布与拉取过程中的安全性与完整性。

签名认证软件的目标用户包括镜像生产者（如镜像开发者，组织团队等）与镜像消费者（如使用镜像的用户）。用户可通过命令交互工具模块与签名系统交互，支持以命令行方式对系统进行操作控制。

容器镜像签名与管理模块是客户端核心功能模块，主要完成镜像签名、签名发布、镜像密钥吊销、自动化签名与验证、信任验证开启与信任信息管理等操作。该模块主要通过调用其他基础模块的能力来完成特定业务操作。

用户通过该模块可对本地构建的镜像进行可信签名，并将签名信息随镜像一同发布到镜像仓库。系统支持对镜像签名状态与信任策略进行统一管理，并支持自动化的签名验证逻辑，提高使用体验。

TUF元数据校验模块负责对签名元数据的格式、完整性、签名以及版本进行逐层验证，保障元信息的合法性与可信性。配套的TUF元数据存储模块用于对签名所需的元文件与密钥进行存储管理，为签名与验证操作提供元数据支撑。

信任仓库与目标文件管理模块实现信任仓库的初始化、同步、内容变更与签名内容列举等功能，是支撑镜像签名可信性的基础。该模块维护着系统的信任链结构与元数据信息，在镜像签名或验证过程中会与服务器侧保持同步。

角色与密钥管理模块负责签名所需密钥的全生命周期管理，包括密钥生成、列举、轮转与吊销等基本操作，并支持元文件签名与委托角色的生成，从而实现基于TUF委托机制的精细化权限控制。

通用模块提供基础性的系统支撑功能，包括签名认证算法、JWS鉴权、配置管理、安全信道通信、日志记录与数据备份等，保障整个签名认证系统的安全性、可靠性与可维护性。其中日志记录功能支持用户操作记录的写入、查询与归档，便于安全审计。

系统在实现形态上，客户端工具作为独立可执行程序提供命令行交互接口，主要用于发起签名、验证与信任仓库管理等操作；服务端则以长期运行服务进程的形式提供元数据管理与存储能力。客户端在执行具体操作时，调用本地API与远程服务协同工作，共同完成镜像签名认证的全过程。

在镜像签名场景中，客户端会根据设定的信任策略调用角色与密钥管理模块完成签名操作，生成的签名元数据通过信任仓库模块进行存储和发布，最终用户在拉取镜像时由客户端自动完成签名验证，确保镜像来源可信、内容完整。通过上述各模块的协同配合，签名认证软件为容器镜像提供了完整的可信生命周期管理能力。

## 外部接口

镜像签名认证软件系统与外部的接口如图所示，主要包括与用户之间的人机交互接口以及集成Docker等容器引擎所提供的镜像安全相关接口。



图3 虚拟机编排调度系统外部接口

（1）与用户之间的人机交互接口

镜像签名认证软件提供命令行工具作为用户与系统交互的主要手段，用于对本地信任库、密钥、签名以及目标镜像进行管理。主要交互命令包括：

1. 信任库初始化与管理相关命令接口

* 信任库初始化，用于初始化信任仓库，并生成TUF相关元数据文件。
* 信任库内容管理，提供列举信任库内容以及对信任库目标文件的增、删、更、查等。
* 元数据生成与签名，提供对生成的相关元数据进行签名的功能。

1. 角色与密钥管理相关命令接口

* 密钥生成与存储，提供生成指定密钥对（如根密钥）的功能，并将私钥加密存储，公钥发布
* 密钥轮换，提供密钥的轮换功能，支持信任策略更新自动轮换、定期自动轮换、时间可配置轮换等多种轮换策略。
* 委托角色管理，提供对委托角色（密钥）的添加、回收、列举等功能。

1. 辅助管理相关命令接口

* 镜像签名认证软件服务器设定功能，支持通过配置文件定义服务端行为。
* 本地密钥文件安全存储信息设定功能。
* 客户端与服务端帮助信息查看功能。

（2）与应用容器引擎之间的接口

镜像签名认证软件系统与 Docker 等应用容器引擎之间通过标准的内容信任接口进行交互。其交互机制如下：

1. 容器镜像签名与管理接口

* 镜像签名创建接口，提供当用户推送镜像时，为镜像创建签名的功能。
* 镜像元数据发布接口，提供推送镜像时，将镜像相关的元数据进行发布的功能。
* 镜像信任信息查看功能，支持查看镜像信任信息，包括相关元数据的详细信息、签名者公钥及其对应密钥ID显示、信任链完整性验证以及密钥层级显示等功能。
* 镜像信任撤销接口，支持撤销指定镜像的信任功能，并支持元数据的修改与版本更新，以及同步服务器端的信任状态。
* 镜像信任验证接口，提供执行镜像拉取时，对拉取的镜像进行信任验证的功能，支持信任链和签名的验证。

1. 环境变量接口

* 镜像信任验证开关接口，提供对镜像信任机制进行开关的功能，开启时强制启用镜像签名认证软件内容信任策略，Docker客户端仅允许推送和拉取其已签名的镜像，且必须通过镜像签名认证软件服务验证签名的有效性；关闭时禁用内容信任，Docker 客户端允许操作未签名镜像，绕过镜像签名认证软件的签名验证流程。
* 内容信任验证所使用的签名认证服务器地址配置接口；
* 配置密钥操作时的密码配置接口。

## 软件部件



图 4 软件部件结构图

软件部件结构如图 4所示，客户端和服务端都包含用户交互工具，信任仓库与密钥管理模块，角色与密钥管理模块，通用模块。同时，客户端用容器镜像签名与管理模块对镜像进行签名验证，服务端通过TUF元数据校验和存储模块，对元数据进行校验和存储。

（1）用户交互命令工具模块：负责接收并识别用户输入的命令种类，然后将命令分发到对应的功能模块来执行。

（2）容器镜像签名与管理模块：该模块是镜像签名认证软件的核心组成部分，专注于容器镜像的签名、发布、信任管理和密钥吊销等关键功能，提供镜像签名认证服务。

（3）TUF元数据校验模块：该模块负责对从信任仓库下载的 TUF 元数据文件进行验证，以确保其有效性和可信度，防止各种安全攻击。

（4）TUF数据存储模块：负责管理镜像签名认证软件在本地存储的 TUF 相关元数据和密钥信息。

（5）信任仓库与目标文件管理模块：负责对信任仓库中的内容进行管理，并提供对目标文件（如容器镜像）对应的元数据文件的相关操作。

（6）角色与密钥管理模块：管理在签名认证过程中涉及的各种角色和密钥，包括密钥的生成、存储、轮换和吊销等。

（7）通用模块：该模块提供一些通用的、被其他模块所依赖的基础功能，以支持整个软件系统的运行和安全。主要包含安全信道通信，JWS鉴权，签名认证算法，日志记录，日志记录，配置管理等子模块。

具体软件部件功能和结构如表 2所示。

表 2 软件部件表

|  |  |  |
| --- | --- | --- |
| 软件单元 | | 说明 |
| 用户交互命令工具 | 命令解析模块 | 对用户的输入命令进行解析匹配 |
| 命令分发模块 | 将命令分发到对应模块执行 |
| 容器镜像签名与管理模块 | 镜像签名 | 提供为容器镜像创建数字签名的功能. 使用目标密钥（Targets Key）对镜像内容摘要进行签名，以确保镜像的完整性和来源可信 |
| 镜像签名发布 | 将镜像的签名元数据发布到信任仓库，使其能够被客户端用于信任验证 |
| 镜像信任信息管理 | 提供查看镜像信任信息的功能. 用户可以查询目标文件的签名元数据的详细信息 |
| 镜像密钥吊销 | 支持撤销对特定镜像的信任，发起吊销请求并生成签名元数据更新，并向服务器端同步信任状态 |
| 信任验证开关 | 提供开启或关闭镜像信任验证功能的设定 |
| 自动化签名与验证 | 支持在推送镜像时自动进行签名，构建信任链，完成签名生成与上传. 同时，在拉取镜像时自动进行信任验证，校验信任链和签名，确保拉取的镜像可信 |
| TUF元数据校验模块 | 格式校验 | 验证 TUF 元数据文件的格式是否符合 TUF 规范 |
| 完整性校验 | 比对元数据文件中记录的哈希值（如 SHA256）与实际下载文件的哈希值，来验证元数据文件在传输过程中是否被篡改 |
| 签名校验 | 使用存储在 root.json 文件中的可信公钥，验证其他元数据文件的数字签名是否有效 |
| 版本校验 | 检查元数据文件的版本号是否符合更新要求，对于 root.json 的更新，还需要进行逐级版本检查 |
| TUF元数据存储模块 | 元文件存储 | 存储服务端的 TUF 元数据文件 |
| 密钥存储 | 安全地存储和管理用于签名和验证的密钥对，主要针对服务端的时间戳密钥和快照密钥 |
| 信任仓库与目标文件管理模块 | 信任库初始化 | 初始化本地信任库，并生成 TUF（The Update Framework）相关元数据文件 |
| 签名内容列举 | 列举信任库内容，查看信任仓库中存储的目标文件信息 |
| 信任库同步 | 与远程信任仓库同步元数据，以保持本地信任信息的最新状态 |
| 信任库内容变更 | 提供对信任库中目标文件的增、删、更、查功能 |
| 角色与密钥管理模块 | 密钥生成 | 生成新的公私钥对的功能，包括根密钥（Root Key）目标密钥（Targets Key），以及服务端使用的时间戳密钥和快照密钥 |
| 密钥列举 | 列举本地存储的所有密钥 |
| 密钥吊销 | 撤销对特定镜像的信任，发起吊销请求并生成签名元数据更新 |
| 密钥轮转 | 提供密钥轮换功能，支持策略变更、临期/逾期告警以及定期自动轮换等机制 |
| 元文件签名 | 对快照元数据生成并签名 |
| 委托角色生成 | 提供委托角色（密钥）添加、回收、列举功能 |
| 通用模块 | 安全信道通信 | 服务端与客户端交互安全通信通道功能 |
| JWS鉴权 | 进行客户端用户鉴权 |
| 签名认证算法 | 提供对多种算法支持和扩展性的功能，例如支持 ECDSA、RSA-PSS 算法 |
| 日志记录 | 提供日志记录的功能 |
| 数据备份 | 提供关键数据的备份恢复功能 |
| 配置管理 | 负责从配置文件或环境变量中读取和解析工作参数 |

各个软件模块调用关系如图 5所示：

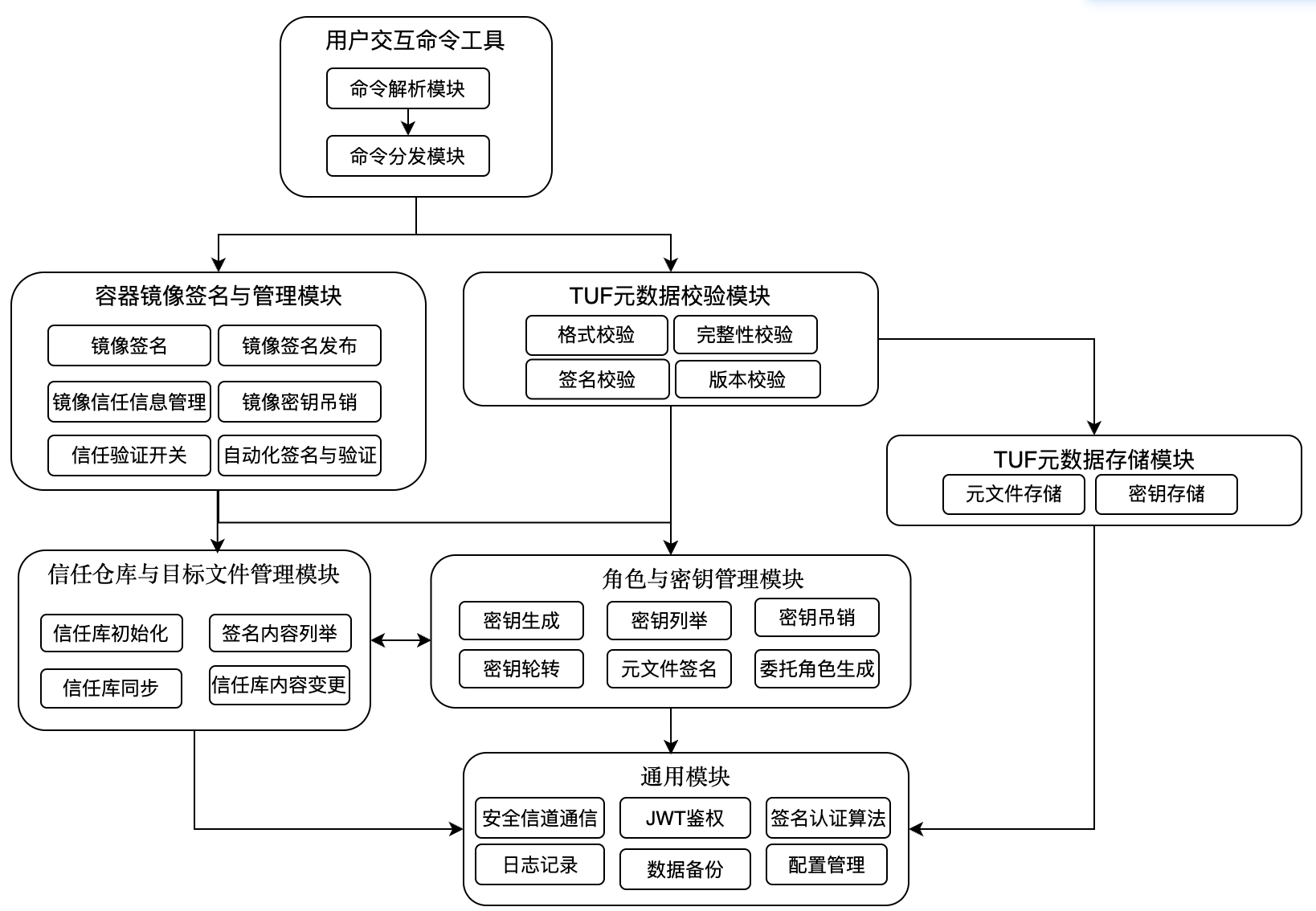


图 5 软件模块调用结构图

当用户上传 (推送) 启用内容信任的容器镜像时，如果信任验证开关被启用，会进行镜像签名认证，客户端需要对信任（仓）库进行初始化，自动生成根密钥（Root Key）目标密钥（Target Key）。私钥需要密钥加密存储，然后使用目标密钥（Targets Key）对镜像标签进行签名。镜像发布者可以自行决定签署哪些标签，签名元数据与镜像的 TAG 部分相关联，最后发布已签名的元数据，包含根元文件，目标元文件。

服务端收到客户端发送的元数据后，对元数据进行校验，然后生成快照密钥和时间戳密钥，保存到密钥数据库存储，之后根据根元文件和目标元文件，生成快照元文件和时间戳元文件，用对应密钥签名后保存到元文件数据库。

当用户拉取启用内容信任的容器镜像时，如果信任验证开关被启用，会进行镜像签名认证，校验元文件的信任链，如果没有问题，则从镜像仓库拉取对应的目标文件，最后检查元文件中的哈希与目标文件的哈希是否一致。

## 技术方案

### 容器镜像签名与管理模块方案

在镜像签名功能中，该模块是容器管理软件的底层加密引擎，主要负责密钥对的生成与管理，包括生成符合密码学标准的 RSA/ECDSA 密钥对（私钥用于签名，公钥用于后续验证），以及将私钥加密存储在本地，确保私钥安全；镜像摘要的签名生成，包括对镜像的 唯一摘要（Digest） 进行哈希（如 SHA-256），以及使用私钥对哈希值进行加密签名，生成符合 JWS（JSON Web Signature） 格式的签名数据；与签名认证软件进行交互，将签名数据和公钥信息上传至签名认证软件服务，由签名认证软件管理签名元数据（如时间戳、密钥版本）。

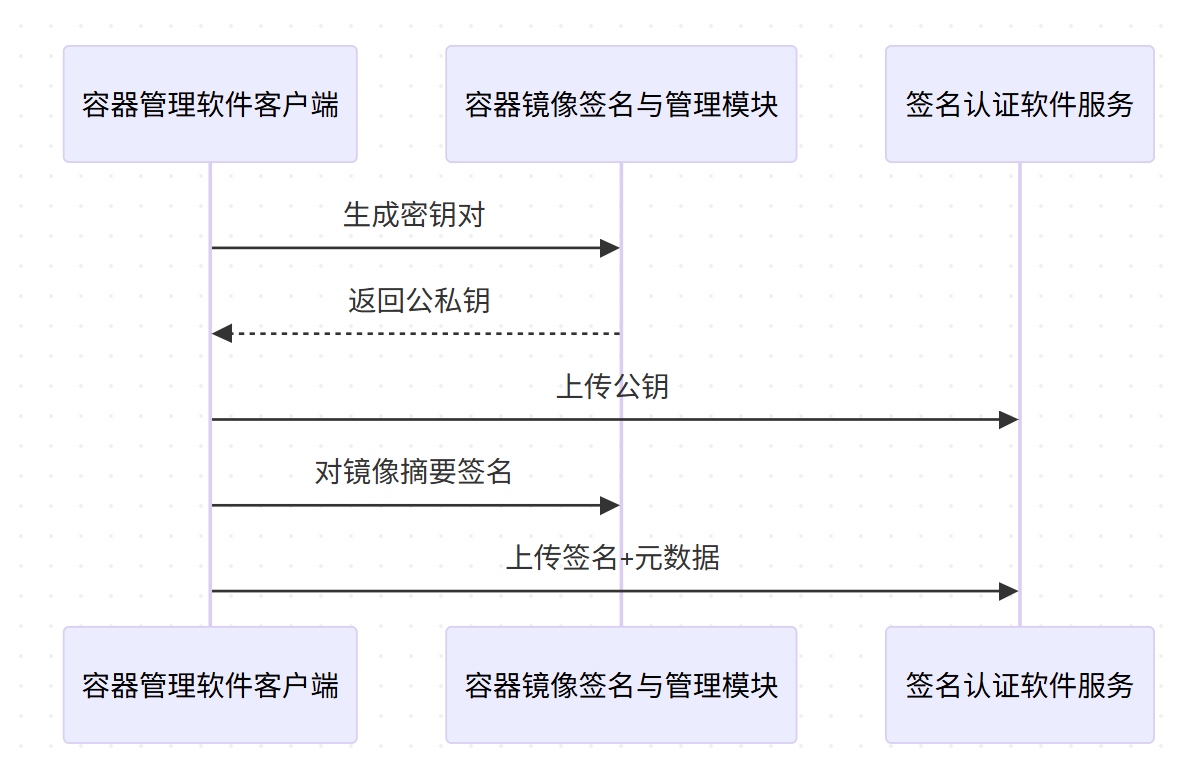


图 6 镜像容器签名工作流程

#### 镜像信任验证开关设定

在镜像信任验证开关设定功能中，该模块负责在验证开关（如 DOCKER\_CONTENT\_TRUST=1）启用后，执行基于密码学的签名验证逻辑，确保镜像的完整性和来源可信性。当用户通过环境变量或配置启用信任验证时，容器管理软件 客户端会在拉取镜像时自动触发内容信任的验证流程：其通过提取镜像元数据中的签名信息，结合本地或远程信任库（签名认证软件服务）存储的公钥，验证签名是否与镜像摘要匹配，同时检查密钥的有效性（如是否过期或被吊销）。该模块不直接管理开关的设定，但负责在开关开启后，提供底层的密码学验证能力，确保只有通过验证的镜像可被加载运行。

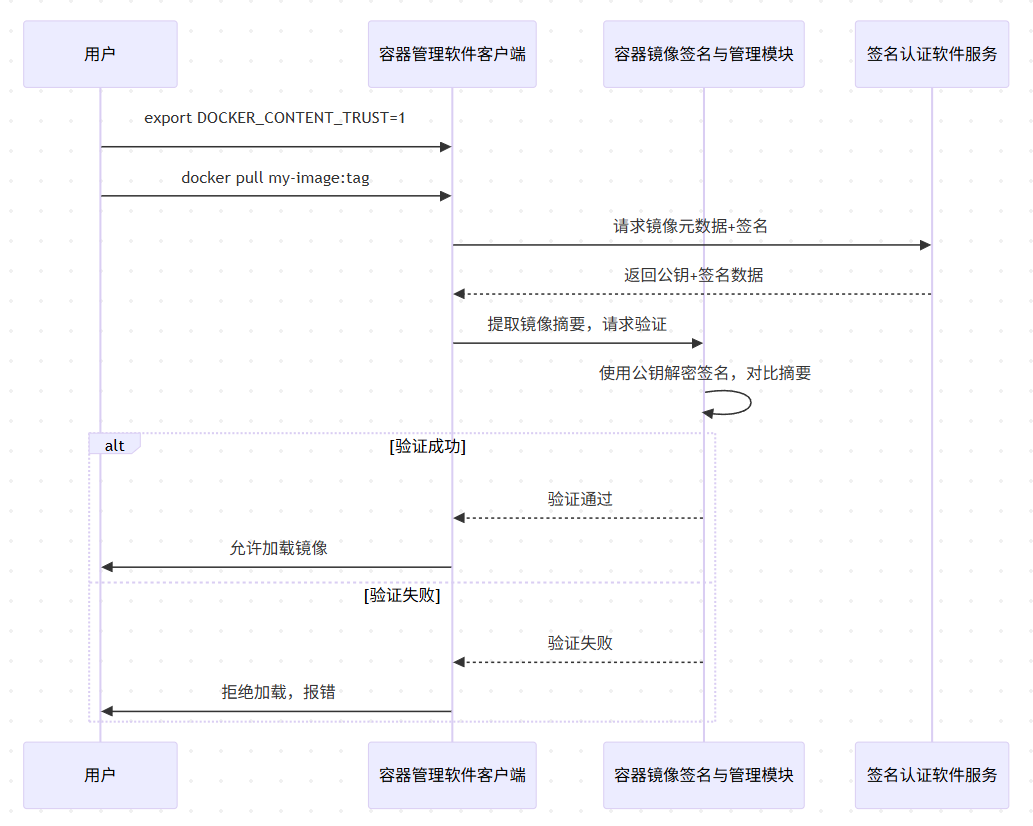


图 7 **镜像信任验证开关设定工作流程**

#### 镜像元数据签名发布

在镜像信镜像元数据发布功能中，该模块负责返回签名秘钥，若不存在生成，并对摘要进行签名，并返回签名数据，以及将元数据、签名和公钥上传至签名认证软件

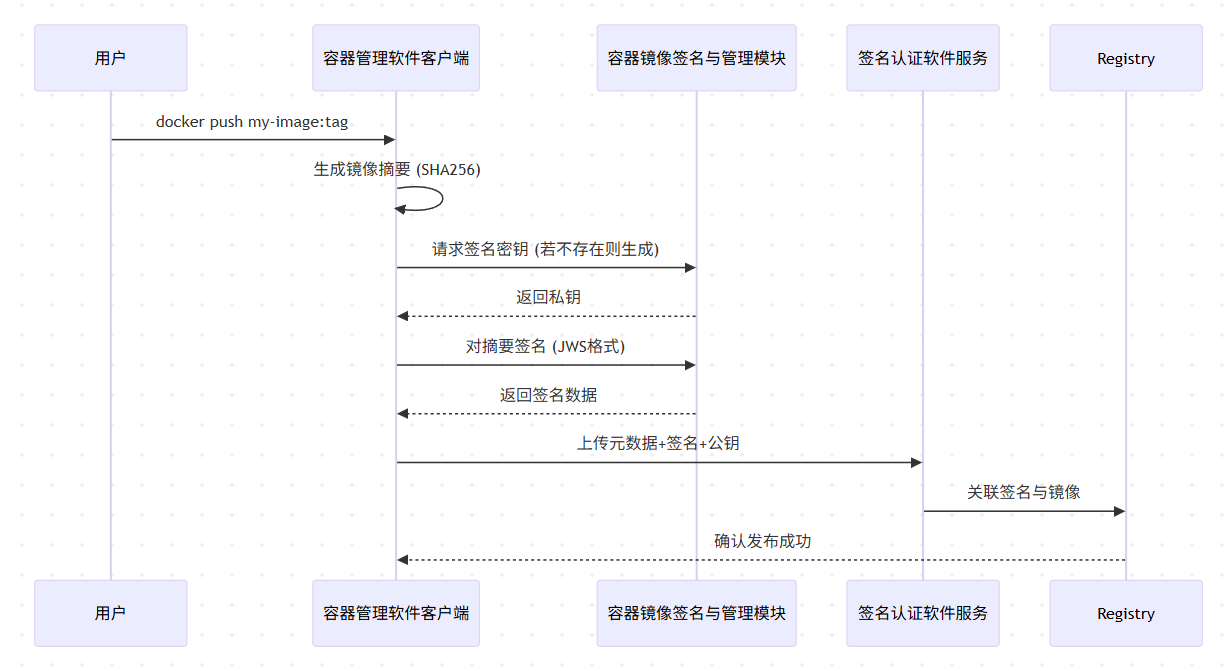


图 8 **镜像元数据签名发布工作流程**

#### 查看镜像信任信息

当在查看镜像信任信息功能中，该模块主要负责解析并展示镜像签名元数据的核心信任信息，确保用户能够透明地验证镜像的签名状态和信任链完整性。当用户通过命令（如 docker trust inspect）查询镜像信任信息时，容器管理软件客户端从 签名认证软件服务拉取存储的签名元数据（包括签名时间、公钥标识、签名算法等），随后该模块对这些原始数据进行解码和格式化处理：其通过解析 JWS（JSON Web Signature）格式的签名数据，提取公钥指纹、签名有效期等关键信息，同时结合本地存储的密钥对状态（如私钥是否存在、公钥是否被吊销），动态生成可读性高的信任报告（如签名者身份、密钥生成时间、关联的镜像层级）。该模块不直接管理元数据存储，但通过密码学验证确保展示的信息未被篡改，从而为用户提供可信的审计依据。

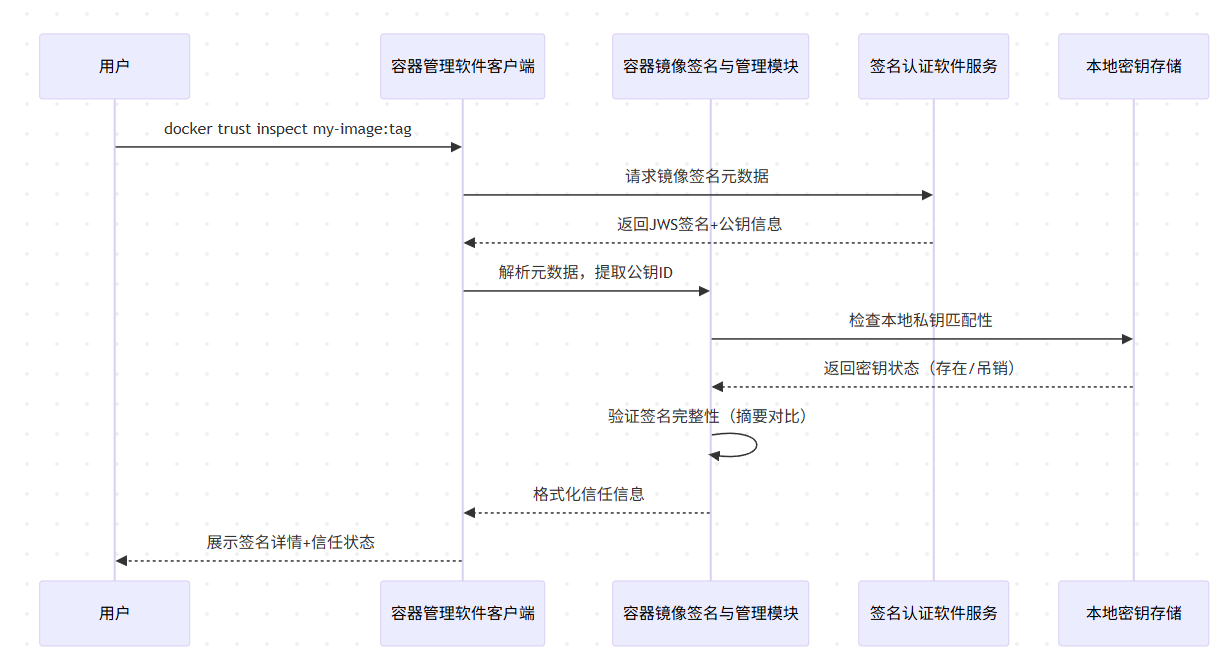


图 9 **查看镜像信任信息工作流程**

#### 撤销指定镜像信任

在撤销指定镜像信任功能中，该模块负责 执行吊销操作的密码学验证与授权凭证生成，确保仅合法密钥持有者可以发起吊销请求并更新信任链。当用户通过 docker trust revoke 命令撤销对某镜像的信任时，容器管理软件客户端会调用 该模块 验证本地私钥与待吊销签名密钥的匹配性（防止未授权操作），随后基于该私钥生成吊销声明的数字签名（JWS 格式），此签名作为吊销请求的授权凭证提交至签名认证软件服务。签名认证软件 服务收到请求后，该模块 协助验证签名合法性，并在信任元数据中标记该镜像签名状态为“已吊销”（如更新 TUF 元数据中的 revoked\_keys 列表），同时触发信任链的版本更新（如递增根元数据的版本号），确保后续镜像拉取操作能检测到吊销状态并阻止使用失效签名。

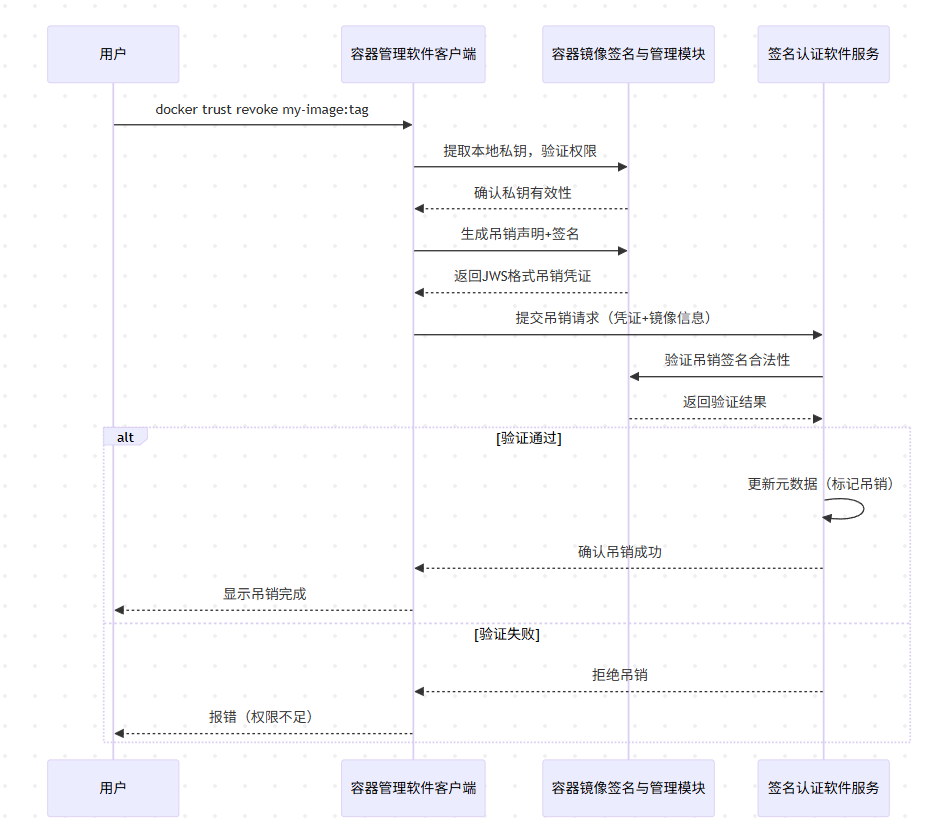


图 10 撤销指定镜像信任工作流程

#### 密钥吊销检查

在密钥吊销检查与元数据更新功能中，该模块 作为 容器管理软件内容信任的核心密码学引擎，主要负责 验证吊销请求的合法性并生成防篡改的吊销凭证，确保密钥吊销操作的安全性和信任链的实时更新。当用户发起吊销请求时，该模块 首先通过本地存储的私钥验证操作者身份（防止未授权吊销），随后基于被吊销密钥的元数据（如密钥 ID、吊销时间戳）生成吊销声明，并使用当前有效的根密钥或委托密钥对该声明进行数字签名（JWS 格式），形成不可抵赖的吊销凭证。此凭证提交至 签名认证软件 服务后，该模块 会协助服务端验证签名有效性，并在验证通过后触发 TUF 元数据的版本化更新（如更新 targets.json 中的吊销列表），同时生成新的根元数据签名以确保信任链的完整性，最终实现全局信任状态的同步，使得所有依赖该密钥的镜像签名自动失效。

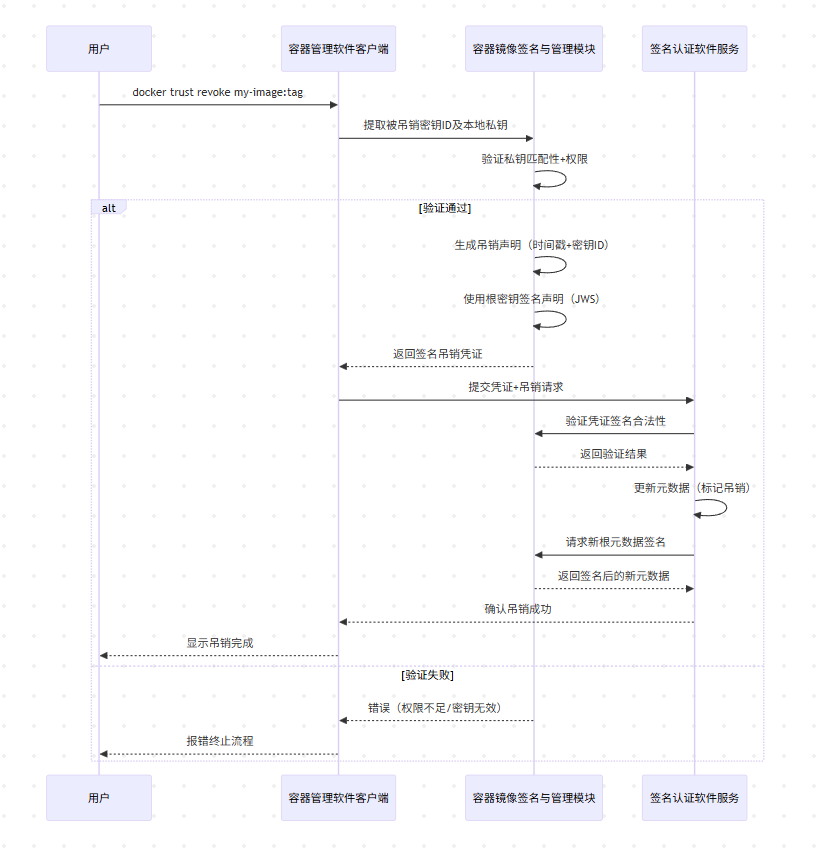


图 11 密钥吊销检查工作流程

#### 推送镜像自动签名

在推送镜像时自动签名的功能中，该模块 作为 容器管理软件内容信任的核心密码学模块，主要负责 在镜像推送过程中无缝触发签名生成与密钥管理，确保镜像内容与发布者身份的不可篡改性。当用户启用内容信任（如设置 DOCKER\_CONTENT\_TRUST=1）并执行 docker push 时，容器管理软件客户端自动调用 该模块 生成或加载本地密钥对，计算镜像各层级（Layer）及清单文件（Manifest）的加密摘要（如 SHA-256），使用私钥对摘要进行签名（JWS 格式），并将签名数据、公钥及时间戳等元数据实时上传至 签名认证软件 服务，形成与镜像绑定的信任链。该模块 在此过程中无需用户显式干预，通过密钥缓存机制避免重复生成密钥，同时确保私钥仅用于本地签名操作，防止泄露风险，最终实现镜像从构建到发布的端到端自动化可信签名。

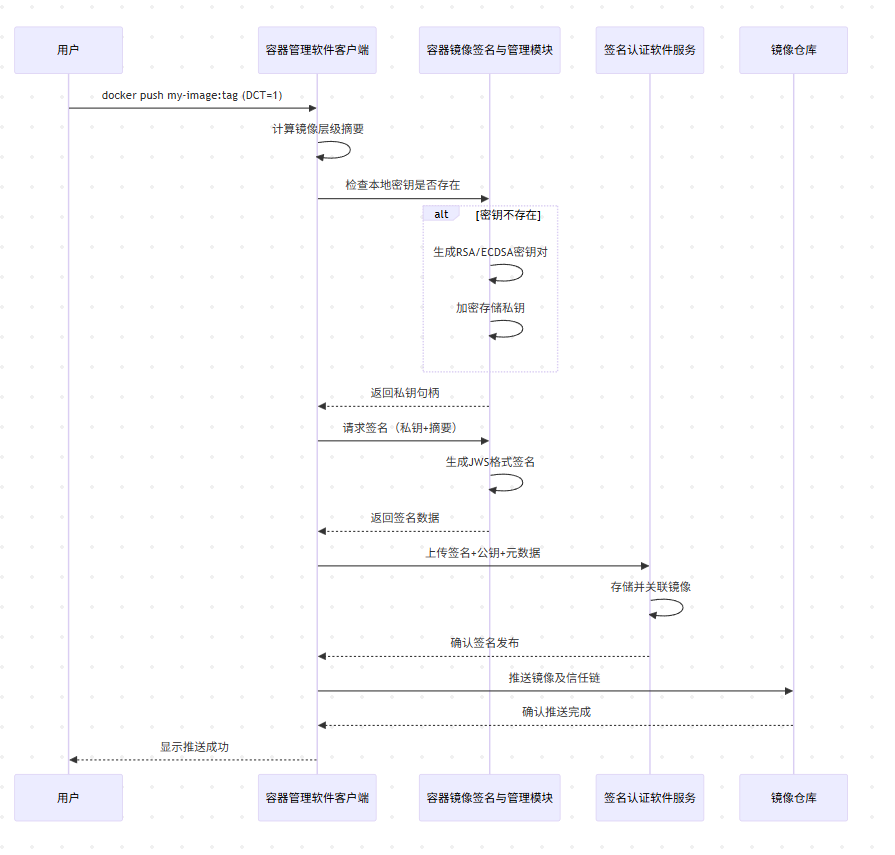


图 12 推送镜像自动签名工作流程

#### 镜像信任验证

在拉取镜像的信任验证功能中，该模块 作为 容器管理软件内容信任的密码学核心，主要负责 解析并验证镜像签名与信任链的完整性和合法性，确保镜像来源可信且未被篡改。当用户启用内容信任（DOCKER\_CONTENT\_TRUST=1）并执行 docker pull 时，容器管理软件客户端从镜像仓库拉取镜像内容的同时，向 签名认证软件 服务请求该镜像的签名元数据（包括签名、公钥、时间戳及信任链层级信息），该模块 基于 TUF（The Update Framework）规范逐级验证信任链的完整性——从根密钥到目标密钥的每一级签名均需通过密码学验证（如验证根元数据是否由可信根密钥签名、目标元数据是否与当前镜像摘要匹配），并检查签名时效性（如是否过期或被吊销）。在此过程中，该模块 依赖本地缓存的根公钥或动态从 签名认证软件 服务获取的元数据，对镜像签名进行解密和摘要比对，若任一环节验证失败（如密钥不匹配、签名被篡改、信任链断裂），则立即阻断镜像加载并向用户返回安全告警，确保仅通过完整信任链验证的镜像可被部署运行。

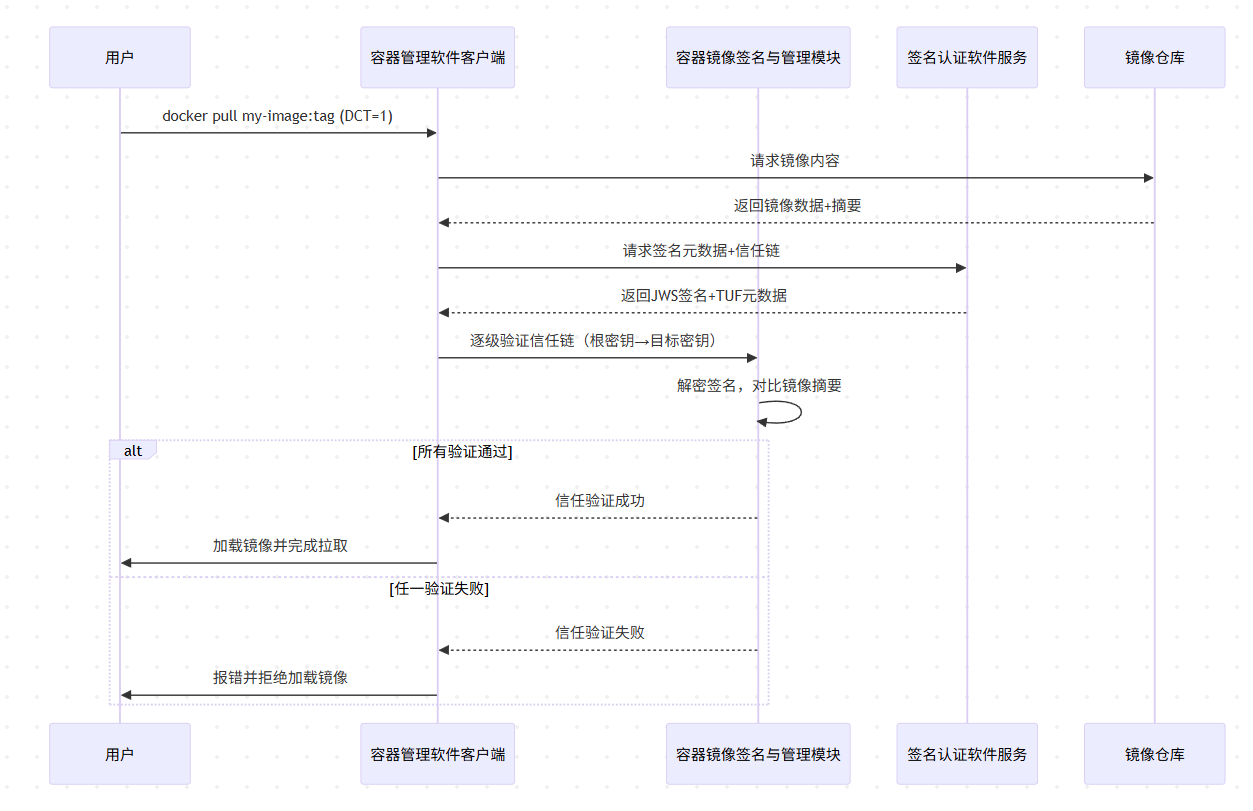


图 13 镜像信任验证工作流程

### TUF元数据校验模块方案

#### TUF角色与元文件定义

本系统中的TUF角色与元文件定义遵循[TUF规范v1.0.33](https://theupdateframework.github.io/specification/v1.0.33/)，各个角色职责如下表所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 角色名称 | 职责概要 | 签名对象 | 密钥位置 | 由谁管理 |
| Root | 管理和分发所有其它角色的公钥。支持密钥轮换、撤销 | Root.json | 离线 | 项目负责人 |
| Targets | 指定哪些镜像（或文件）被信任，可发布。支持 Delegation | Targets.json | 离线 | 项目负责人 |
| Snapshot | 确保 targets（及其 delegation）元数据的一致性与版本管理，防止回滚和混合匹配攻击 | Target.json及其Delegation | 可存服务端 | 自动化构建系统或服务器 |
| Timestamp | 指出最新的 snapshot 版本，减轻冻结攻击 | Snapshot.json | 服务端 | 自动化构建系统或服务器 |

不同角色与元文件的关系定义如图 6所示：



图 14 角色与元文件关系图

#### TUF元数据校验

1. 格式校验：TUF 规范允许实现者选择任何数据格式元数据文件，但要求所有规范中定义的字段都必须包含且能够被客户端无歧义地解析，推荐使用支持规范化（canonicalization）的数据格式，如 JSON 的子集。本模块采用的具体元数据格式 JSON进行格式校验，确保文件结构符合预期，并且包含所有必要的字段，如 \_type, spec\_version, version, expires, meta, signatures, signed 等。
2. 完整性校验：为了防止中间人攻击者的混合搭配攻击，需要对下载的元数据文件的完整性进行校验。

* Snapshot 元数据 文件中会记录 Targets 元数据 及其 delegation 的哈希值，在更新 Snapshot 元数据时，客户端需要校验其哈希值是否与 Timestamp 元数据中记录的哈希值一致。
* Timestamp 元数据 文件会记录 Snapshot 元数据 的哈希值。在更新 Targets 元数据时，需要校验其哈希值是否与 Snapshot 元数据中记录的哈希值一致。
* 这种哈希校验机制可以确保元数据文件在传输过程中没有被篡改。

1. 签名校验：TUF 的核心安全机制之一是使用数字签名来验证元数据文件的真实性。 每个元数据文件（Root, Timestamp, Snapshot, Targets）都由一个或多个与相应角色关联的密钥进行签名。 Root 元数据 文件中包含了所有其他角色的公钥。 在更新任何元数据文件时，客户端必须使用可信的 Root 元数据中记录的公钥来验证其签名，每个参与签名的密钥 ID (KEYID) 在计算阈值时只能贡献一个签名。 如果签名验证失败，客户端必须丢弃该元数据文件并中止更新
2. 版本校验：TUF 使用版本号来防止回滚攻击。 每个元数据文件都有一个版本号。 在更新元数据文件时，客户端必须检查新文件的版本号是否大于或等于旧文件的版本号

图 7说明了校验流程使用了元文件的哪些字段。



图 15 元文件校验字段

### 角色与密钥管理模块方案

#### （1）客户端密钥生成与管理

密钥生成模块接收角色类型与算法参数（如ECDSA，RSA等），生成不同角色（如 Root, Targets, Snapshot, Timestamp）的公私钥，符合 TUF 规范的密钥对。

对于客户端而言，主要生成的是根密钥和目标密钥，生成后的私钥通过加密方式本地存储，用户需通过交互或环境变量设置密钥解密密码。

对于服务端而言，主要生成的是时间戳密钥和快照密钥，生成后的私钥存储在密钥库中加密保存，用以后续服务器进行时间戳和快照元文件签名。

支持密钥列举接口，返回密钥仓库中所有已注册密钥的角色、KeyID、创建时间等元信息。

#### （2）元文件签名

该模块负责使用与每个角色关联的私钥对相应的元数据文件进行签名。

Targets 角色使用其私钥对 targets.json 进行签名，Snapshot 角色使用其私钥对 snapshot.json 进行签名，Timestamp 角色使用其私钥对 timestamp.json 进行签名，Root 角色使用其私钥对 root.json 进行签名。

对于客户端，需要负责Root和Target角色的签名。

对于服务端，需要负责Timestamp角色的签名。

Snapshot角色在哪里签名，取决于系统配置，可以在客户端生产Snapshot密钥并签名，也可以选择把Snapshot密钥托管到服务端，让Snapshot在服务端签名。

#### 元文件更新

元文件格式采用[TUF规范v1.0.33](https://theupdateframework.github.io/specification/v1.0.33/)中定义的格式，在不再赘述，不同角色的元文件更新流程如下。

**更新Root.json：**

Root.json更新需要逐级建立信任链，也就是从版本1，逐级更新到版本N，中间不能跳跃。

1. 令N表示当前可信根元数据文件的版本号。

2. 尝试下载Root.json的N+1版本，最多X字节

3. 检查是否存在恶意软件攻击：

1. 需要超过Threshold个版本N的Root.json签名验证通过（垂直验证）

2. 需要超过Threshold个版本N+1的Root.json签名验证通过（水平验证）

4. 检查是否存在回滚攻击：

1. 新Root.json版本号 = 旧Root.json版本号 + 1

5. 检查是否存在冻结攻击：中间Root.json过期没有关系

6. 持久化Root.json

7. 重复2～9步

8. 检查是否存在冻结攻击：最终的Root.json文件过期时间戳未到，且高于固定更新时间

9. 持久化最新Root.json

**更新Timestamp.json**

1. 下载Timestamp.json，最多X字节

2. 检查是否存在恶意软件攻击：用Root.json记录的公钥验证Timestamp.json签名，需要超过Threshold个签名验证通过

3. 检查是否存在回滚攻击：

1. 新Timestamp.json版本号 > 旧Timestamp.json版本号

2. 新Timestamp.json中记录的Snapshot.json版本号 >= 旧Timestamp.json中记录的Snapshot.json版本号

4. 检查是否存在冻结攻击：新TimeStamp.json文件过期时间戳未到，且高于固定更新时间

5. 持久化Timestamp.json

**更新Snapshot.json**

1. 下载Snapshot.json，最多X字节

2. 防止中间人攻击者的混合搭配攻击：校验新Snapshot.json的哈希值是否与Timestamp.json记录的哈希值一致

3. 检查是否存在恶意软件攻击：用Root.json记录的公钥验证Snapshot.json签名，需要超过Threshold个签名验证通过

4. 检查是否存在回滚攻击：

1. 新Snapshot.json版本号 = 最新的Timestamp.json中记录的Snapshot.json版本号

2. 新Snapshot.json中记录的Target.json版本号 >= 旧Snapshot.json中记录的Target.json版本号

5. 检查是否存在冻结攻击：新Snapshot.json文件过期时间戳未到，且高于固定更新时间

6. 持久化Snapshot.json

**更新Target.json**

1. 下载Target.json，最多X字节

2. 防止中间人攻击者的混合搭配攻击：校验新Target.json的哈希值是否与Snapshot.json记录的哈希值一致

3. 检查是否存在恶意软件攻击：用Root.json记录的公钥验证Target.json签名，需要超过Threshold个签名验证通过

4. 检查是否存在回滚攻击：

1. 新Target.json版本号 = 最新的Snapshot.json中记录的Target.json版本号

5. 检查是否存在冻结攻击：新Target.json文件过期时间戳未到，且高于固定更新时间

6. 持久化Target.json

#### 密钥吊销与轮换

当密钥泄露或不再使用时，需要将其吊销，密钥吊销后，客户端将不再信任使用被吊销密钥签名的任何元数据。

提供灵活的密钥轮换机制，支持按策略、周期、角色分类轮换密钥，确保长期安全性。

密钥轮换场景：

* 策略轮换：监听信任策略更新事件（如角色变更），触发相关角色密钥轮换；
* 告警轮换：当密钥临近过期或已过期时，接口返回轮换告警，客户端可触发轮换流程；
* 定期轮换：接口支持通过配置参数设定轮换周期，到期后自动轮换；
* 分层轮换：按角色进行分层轮换，允许针对Root更严格轮换策略；

吊销密钥的操作是通过更新上层角色（对于顶层角色来说是Root，对于委托角色来说是上层的委托角色）元数据文件来实现的，主要步骤如下。

1. 使用密钥生成功能生成新密钥并保存
2. 将需要吊销或轮转的公钥从上层元数据文件中移除，添加新密钥的公钥
3. 使用上层角色的私钥重新签名发布新的上层元数据文件
4. 使用新密钥的私钥重新签名当前元数据文件
5. 级联更新其他上层元数据（比如上层targets.json更新后，对应的snapshot.json和target.json也需要更新）
6. 同步到服务端



图 16 密钥吊销工作流程

#### 委托角色生成

TUF 支持 角色委托 (Role Delegation)，允许一个角色（通常是 Targets 角色）将对特定目标文件的信任委托给另一个角色。

主要方式是通过在 targets.json 文件中指定 delegation 信息和被委托角色的可信密钥。



图 17 生成委托角色工作流程

### 信任仓库与目标文件管理模块

#### 信任仓库初始化

信任库的初始化涉及到生成初始的 TUF 元数据文件和密钥，客户端生成Root，Target角色的密钥与元文件，服务端生成Snapshot，Timestamp角色的密钥与元文件，客户端与服务端进行同步，具体流程如下：

1. 客户端与服务端之间用JWS 令牌进行身份验证，确保客户端有权初始化信任库
2. 客户端使用密钥生成功能，生成Root和Target密钥，加密保存在本地存储中
3. 客户端使用Root和Target密钥，生成Root.json，Target.json，保存在本地文件缓存，并发送给服务端
4. 服务端使用密钥生成功能，生成Snapshot和Timestamp角色的密钥，加密保存在密钥数据库中
5. 服务端使用Snapshot和Timestamp密钥，结合收到的Root.json和Target.json，生成Snapshot.json，Timestamp.json
6. 服务端收到的Root.json，Target.json，Snapshot.json，Timestamp.json，都保存到元文件数据中。
7. 服务器不会向客户端主动推送元文件
8. 当客户端拉取元文件时，用本地的Root.json对收到元文件进行校验，校验无误后，保存最新状态的元文件到本地文件缓存

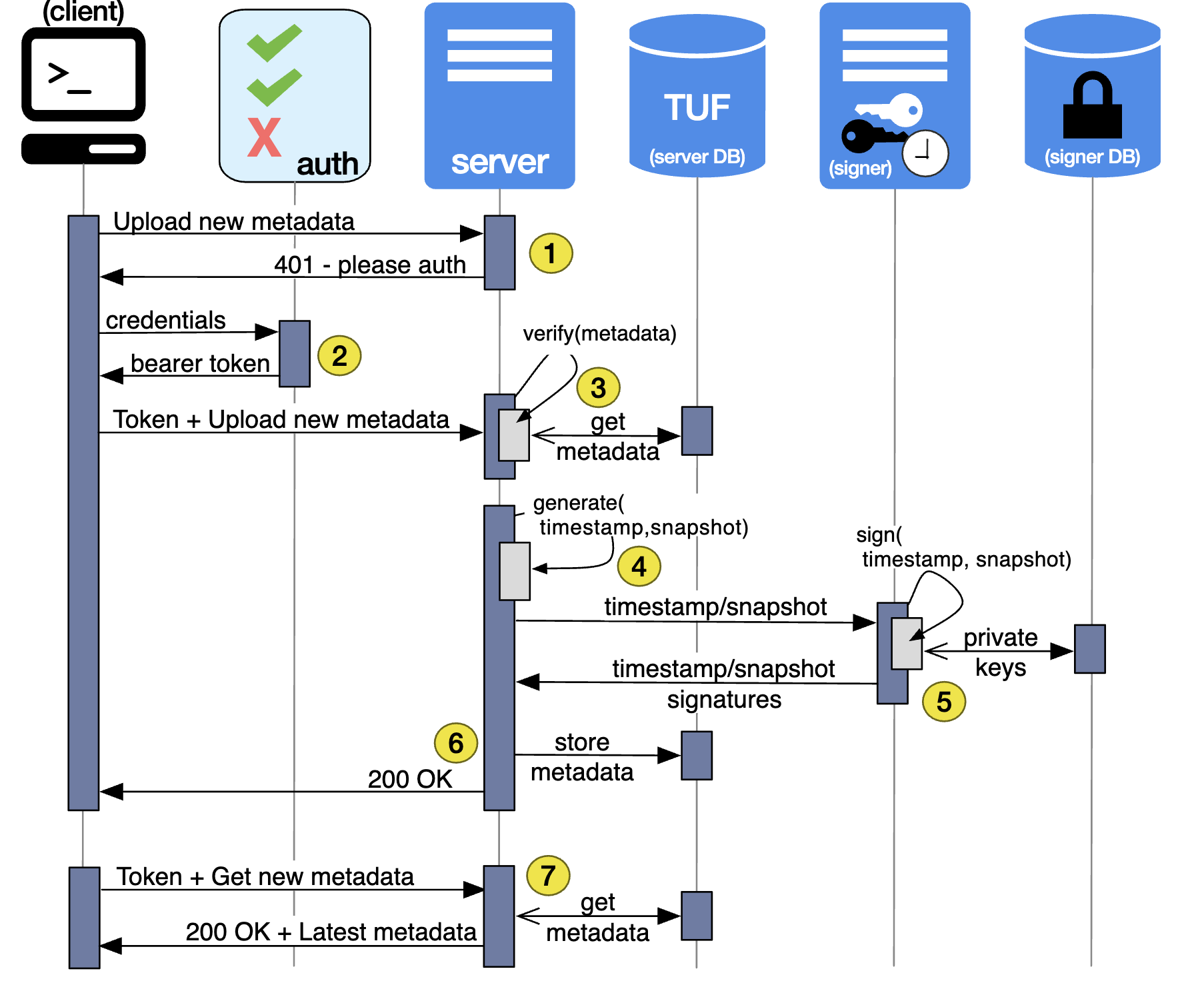


图 18 信任仓库初始化工作流程

#### 信任仓库内容变更

支持对信任库中目标文件的增、删、更、查操作。客户端更新 targets.json 元数据文件，并重新签名上层对target.json，如果涉及到更新顶层target.json文件，那么还会更新root.json。服务器端接收客户端的这些变更请求，进行验证和处理，并更新snapshot.json和timestamp.json文件。

新增目标文件：

1. 使用密钥生成功能，生成新目标密钥
2. 对目标文件进行哈希，得到哈希值
3. 生成目标文件的target.json，记录目标文件哈希值，并用新密钥加密
4. 进行路径匹配，找到改目标文件所处路径的上层角色，在上层角色的json文件中添加新公钥，必要时需要更新root.json
5. 将更新后的元文件发送到服务端
6. 服务端根据新target.json，更新的snapshot.json和timestamp.json



图 19 信任仓库内容变更工作流程

修改目标文件：

1. 对目标文件机械能哈希，得到哈希值
2. 更改目标文件target.json中的哈希值，重新签名
3. 将更新后的元文件发送到服务端
4. 服务端根据新target.json，更新的snapshot.json和timestamp.json

删除目标文件：

1. 删除目标文件的target.json
2. 更新上层元文件，删除上层元文件中，该target.json对应的公钥
3. 将删除请求和更新后的元文件发送到服务端
4. 服务端删除目标文件的target.json，更新上层元文件，更新snapshot.json和timestamp.json

查找目标文件：

1. 拉取服务端最新的元文件数据
2. 进行路径匹配，找到目标文件对应的target.json

### 通用模块技术方案

通用模块作为签名上传系统的基础支撑模块，负责系统的服务入口、加密算法实现、运行配置管理及日志审计与备份等核心通用功能，贯穿系统的各个子系统与服务模块。该模块以模块化、解耦化为设计原则，划分为安全信道通信子模块、签名认证算法子模块、配置管理子模块、日志记录子模块和数据备份子模块五个部分，分别提供外部通信接口、加密服务支持、全局配置管理以及系统运行和容灾保障。

通用模块的整体目标是为系统提供稳定、高可用、可审计、易维护的运行基础环境，支撑签名上传、验证、元数据处理、密钥管理等核心业务流程的顺利执行。



**图 20 通用模块功能划分**

#### 安全信道通信子模块

安全信道通信模块作为镜像签名认证软件客户端与服务器端的桥梁，主要用于客户端与服务器的安全通信以及元数据的安全传输，负责建立客户端与服务器端的安全通信通道，支持客户端与服务器端之间的认证与加密传输，并为系统提供统一的错误重试、连接管理机制，提高可靠性与容错能力。其在系统中的位置如下图所示。



**图 21 安全信道通信模块系统功能**

镜像签名认证软件启动时，系统会对安全信道通信模块进行初始化，进行握手与认证，客户端建立与服务端的HTTPS/TLS安全连接，执行身份认证，建立安全通信通道。

客户端将签名后的 TUF 元数据（如 root.json, targets.json, snapshot.json, timestamp.json）通过 POST/PUT 请求或RPC请求发送至服务器时，安全信道通信模块负责进行必要的签名封装（如 JWS）与 HTTP Header 加密设置，服务端安全信道通信模块接收请求并转发至 TUF 校验逻辑模块，服务端校验成功后写入后端存储并返回 200/ACK 响应，若请求失败，则返回错误码，由客户端安全信道通信模块进行错误处理或重试。

当用户拉取镜像时，客户端会通过安全信道通信模块向服务端安全信道通信模块发起元数据请求，服务端返回 TUF 元数据，安全信道通信模块负责校验数据完整性与签名有效性，客户端据此决定是否允许镜像拉取操作。

同时安全信道通信模块还支持网络异常重试、连接超时与断线重连、错误日志记录以及自定义错误响应类型等错误处理机制。

#### 签名认证算法子模块

签名认证算法模块是整个镜像签名与验证系统的核心安全基础，为签名生成、签名验证、公私钥管理、加密封装等多种功能提供基础支持，并提供加密算法多样性适配。其在系统中的功能如下图所示。



**图 22 签名认证算法模块系统功能**

用户推送镜像或元数据进行签名时，会接收元数据结构并提取签名角色及其私钥，之后调用对应算法（如ECDSA/RSA-PSS）进行签名，并根据用户配置封装为JWS格式，之后返回签名后的数据由安全信道通信模块进行上传。

用户因拉取镜像而需要对签名进行验证时，系统提取元数据 JSON 和签名字段，从本地或根元数据提取相应公钥，根据签名算法校验签名是否合法，返回验证结果，通知客户端是否允许拉取镜像。

签名认证算法模块提供多种签名认证算法，包括ECDSA、RSA-PSS算法，并支持签名算法的扩展，用户可使用私有的签名算法进行替换；该模块还提供JWS格式的签名封装算法支持。

#### 配置管理子模块

配置管理模块负责系统中可变参数和运行配置项的统一管理，确保各功能模块能够灵活、可控地获取运行所需的配置信息。提供统一配置读取与分发、配置校验与默认值处理、服务端行为定义配置以及敏感配置项保护（如私钥存储路径、访问令牌等）功能。模块整体工作流程如下图所示。



**图 23 配置管理模块工作流程**

系统启动时会对配置管理模块进行初始化，加载配置文件、读取环境变量并应用配置默认值，之后会对关键配置项进行合法性与完整性校验，包括密钥路径是否存在、密钥密码是否为空、用户指定的加密算法是否合法，并对配置格式进行验证，之后缓存配置项供其他模块使用。

用户修改配置项时，首先通过客户端输入修改参数或修改配置文件提交配置变更请求，配置管理模块会重新加载并校验变更项，对需要重启系统的配置项进行用户提示输出，最后将修改后的配置项广播到依赖模块（如加密模块与签名模块）实现配置项更新。

用户可以通过配置管理模块配置镜像信任验证开关，当该项开启时，强制启用镜像签名认证软件内容信任策略，Docker客户端仅允许推送和拉取其已签名的镜像，且必须通过镜像签名认证软件服务验证签名的有效性；该项关闭时禁用内容信任，Docker 客户端允许操作未签名镜像，绕过镜像签名认证软件的签名验证流程。同时还支持用户进行密钥密码配置、密钥轮换策略配置、加密算法配置以及私钥存储路径配置等多种配置项，并支持通过配置文件定义服务端行为。

#### 日志记录子模块

日志记录子模块负责记录系统运行、用户操作、签名请求、配置变更、安全事件等，便于审计与追踪问题。

日志记录子模块支持多种类型的日志记录，包括系统启动日志（系统启动、配置加载、模块初始化等），操作审计日志（用户登录、配置修改、签名请求、信任验证过程等），错误日志（签名失败、配置错误、通信失败、安全检查不通过等）以及安全事件日志（密钥轮换、认证失败、非法请求、访问敏感资源等），并支持DEBUG / INFO / WARN / ERROR多种级别日志输出。

#### 数据备份子模块

数据备份子模块负责对 TUF 元数据、密钥文件、重要配置等进行备份，保障系统安全性与可恢复能力。

数据备份子模块支持对关键数据的备份与恢复功能，提供本地压缩打包以及远程上传的备份方式，可通过定时任务或手动触发，备份过程中会提供元数据的一致性保障，避免文件写入中断，并对密钥与配置等关键数据的备份进行加密保护，系统还提供恢复脚本用于快速恢复的支持。

## 内部接口设计

### 容器镜像签名与管理模块接口

|  |  |  |
| --- | --- | --- |
| 功能 | 接口名称 | 接口功能 |
| 密钥管理 | GenerateKeyPair() | 生成 RSA/ECDSA 密钥对（公钥和私钥），支持密钥长度和算法选择。 |
| LoadPrivateKey() | 从本地加密存储加载私钥，需用户密码解密。 |
| SavePrivateKey() | 将私钥加密后存储到本地文件系统，防止未授权访问。 |
| 签名生成与验证 | SignData() | 使用私钥对镜像摘要进行签名，生成符合 JWS 标准的签名数据。 |
| VerifySignature() | 使用公钥验证签名与镜像摘要的匹配性，返回布尔值表示验证结果。 |
| ParseJWS() | 解析 JWS 格式的签名数据，提取算法、公钥 ID 和签名体。 |
| GenerateTimestampToken() | 生成带时间戳的签名令牌，防止重放攻击。 |
| 元数据操作 | BuildMetadata() | 构建镜像元数据（如 Manifest 的 JSON 结构），包含层级摘要和标签信息。 |
| SignMetadata() | 对元数据文件进行签名，生成签名块并关联到 TUF 元数据中。 |
| UpdateRevocationList() | 在元数据中更新吊销列表（如 revoked\_keys），标记失效密钥。 |
| 信任链管理 | ValidateTrustChain() | 构建镜像元数据（如 Manifest 的 JSON 结构），包含层级摘要和标签信息。 |
| FetchRootMetadata() | 对元数据文件进行签名，生成签名块并关联到 TUF 元数据中。 |
| CheckKeyExpiration() | 检查密钥有效期，触发告警或自动吊销过期密钥。 |
| 吊销相关 | GenerateRevocationClaim() | 生成吊销声明（包含吊销时间、密钥 ID），并使用根密钥签名。 |
| ValidateRevocation() | 验证吊销请求的签名合法性，确保仅授权用户可执行吊销操作。 |
| PropagateRevocation() | 将吊销状态同步到所有依赖的镜像元数据中，更新信任链版本。 |
| 辅助工具 | EncryptPrivateKey() | 使用 PBKDF2 或 AES 加密私钥，保护本地存储安全。 |
| DecodeX509Certificate() | 解析 X.509 证书链，提取公钥和颁发者信息。 |
| GenerateNonce() | 生成随机数（Nonce），增强签名唯一性，防止重放攻击。 |
| 错误处理 | NewTrustError() | 生成标准化的信任错误对象，包含错误码和上下文信息。 |
| LogSecurityEvent() | 记录安全事件（如签名失败、密钥吊销），支持审计和告警。 |

### 角色与密钥管理模块接口

|  |  |  |
| --- | --- | --- |
| 功能 | 接口名称 | 接口功能 |
| 密钥管理 | KeyGenerate() | 生成新密钥，用以元文件签名，可指定密钥对应的角色 |
| KeyList() | 列出当前存储的密钥，包含信息：密钥id，角色，存储位置等 |
| KeyRemove() | 移除某个密钥，删除对应的密钥文件 |
| KeyPasswd() | 改变某个密钥的加密密码 |
| KeyRotate() | 轮转某个密钥 |
| KeyExport() | 导出密钥 |
| KeyImport() | 倒入密钥 |
| 角色管理 | DelegationAdd() | 添加角色委托 |
| DelegationList() | 列举所有委托角色，包含信息：角色名称，匹配的路径等 |
| DelegationRemove() | 移除角色委托 |

### 信任仓库与目标文件管理模块接口

|  |  |  |
| --- | --- | --- |
| 功能 | 接口名称 | 接口功能 |
| 信任仓库管理 | RepositoryInit() | 初始化信任仓库。如果本地没有密钥，自动生成根密钥，目标文件密钥。生成root.json，target.json并上传到服务器 |
| RepositoryDelete() | 移除某个信任仓库，删除该仓库的密钥和元文件 |
| 仓库同步管理 | RepositoryPublish() | 将本地仓库的修改同步到服务器中对应的远端仓库 |
| RepositoryStatus() | 查看当前本地仓库的修改 |
| RepositoryReset() | 撤销本地仓库的修改 |
| 目标文件管理 | RepositoryAdd() | 向某个信任仓库中添加目标文件 |
| RepositoryRemove() | 从某个信任仓库中移除某个目标文件 |
| RepositoryList() | 拉取远程仓库的最新状态，列出仓库中目标文件信息 |

### 通用模块接口

表2‑3通用模块内部接口

|  |  |  |
| --- | --- | --- |
| 功能 | 接口名称 | 接口功能 |
| 安全信道通信 | HandleClientRequest() | 统一接收客户端请求，如签名上传、镜像推送、验证等，进行解析与初步处理 |
| AuthenticateUser() | 对请求用户进行身份鉴权，判断其角色与权限 |
| AuthorizeKeyUsage() | 将解析后的请求分发至内部相关模块 |
| RespondToClient() | 将处理结果封装为响应信息返回客户端 |
| 签名认证算法 | SignContent() | 对给定内容进行数字签名，支持算法如 RSA-PSS、ECDSA |
| VerifySignature() | 验证数字签名是否合法，支持多种算法与安全参数 |
| EncryptData() | 对输入数据进行加密处理 |
| DecryptData() | 对输入数据进行解密处理 |
| ComputeDigest() | 计算输入数据的摘要值 |
| LoadCryptoConfig() | 从配置模块加载当前加密算法及参数策略 |
| RefreshKeyCache() | 密钥变更或轮换后刷新密钥缓存 |
| 配置管理 | LoadConfigFromFile() | 从配置文件中加载系统初始化配置信息 |
| LoadConfigFromEnv() | 从环境变量中加载配置信息 |
| GetConfigValue() | 获取指定配置项的当前值 |
| GetModuleConfig() | 获取指定模块的完整配置信息 |
| SetConfigValue() | 更新某项配置，并触发模块热更新 |
| ReloadModuleConfig() | 通知模块重新加载配置（如算法切换等场景） |
| GetKeyRotationPolicy() | 获取密钥轮换策略配置 |
| GetSignaturePolicy() | 获取签名算法策略与密钥参数配置 |
| GetStoragePolicy() | 获取密钥与元数据的存储路径与访问配置 |
| GetRepoSyncPolicy() | 获取信任仓库同步策略与相关配置 |
| 日志记录 | LogInfo() | 记录系统运行过程中的普通信息日志 |
| LogWarn() | 记录警告类日志，如配置异常、响应超时等 |
| LogError() | 记录错误日志，如模块故障、签名失败、非法访问等 |
| QueryLog() | 按条件查询日志记录，支持关键字、时间范围、日志级别筛选 |
| 数据备份 | BackupMetadata() | 备份当前元数据状态，防止系统故障时丢失 |
| BackupKeyStorage() | 备份密钥文件目录，支持周期性自动备份 |
| RestoreFromBackup() | 从指定备份恢复元数据或密钥文件 |
| ScheduleBackupJob() | 设置备份任务调度策略，如定时、触发备份等 |
| ValidateBackupFile() | 校验备份文件完整性，确保可用于恢复 |

# 测试与验证策略

## 单元测试策略

（1）测试框架

单元测试框架采用Google Test (gTest)，该框架是C++社区中最流行的测试框架之一。

gTest框架提供了丰富的断言机制，如 EXPECT\_EQ、ASSERT\_TRUE 等，可以方便地检查各种条件。

gTest框架提供了Mock支持，与 Google Mock 配合使用，可以进行模拟测试（mocking），模拟外部依赖。

gTest框架支持自定义测试过滤器、测试事件和其他扩展功能。

gTest框架跨平台能力好，支持多种平台，包括 Linux和Windows。

（2）单元测试流程

a. 将所研发软件系统划分成清晰的模块：将被测系统的功能划分为多个清晰的模块，例如 Chart 包管理、云应用管理、仓库管理模块等。每个模块都有单独的职责，便于单元测试。

b. 编写单元测试用例：在Google Test中，每个参与测试用例通过TEST宏定义。每个测试用例通过Google Test提供的丰富的断言用来检查被测试的C++函数或模块的返回值是否符合预期。

c. 构建和运行测试：使用CMake编译和构建项目，然后运行测试。

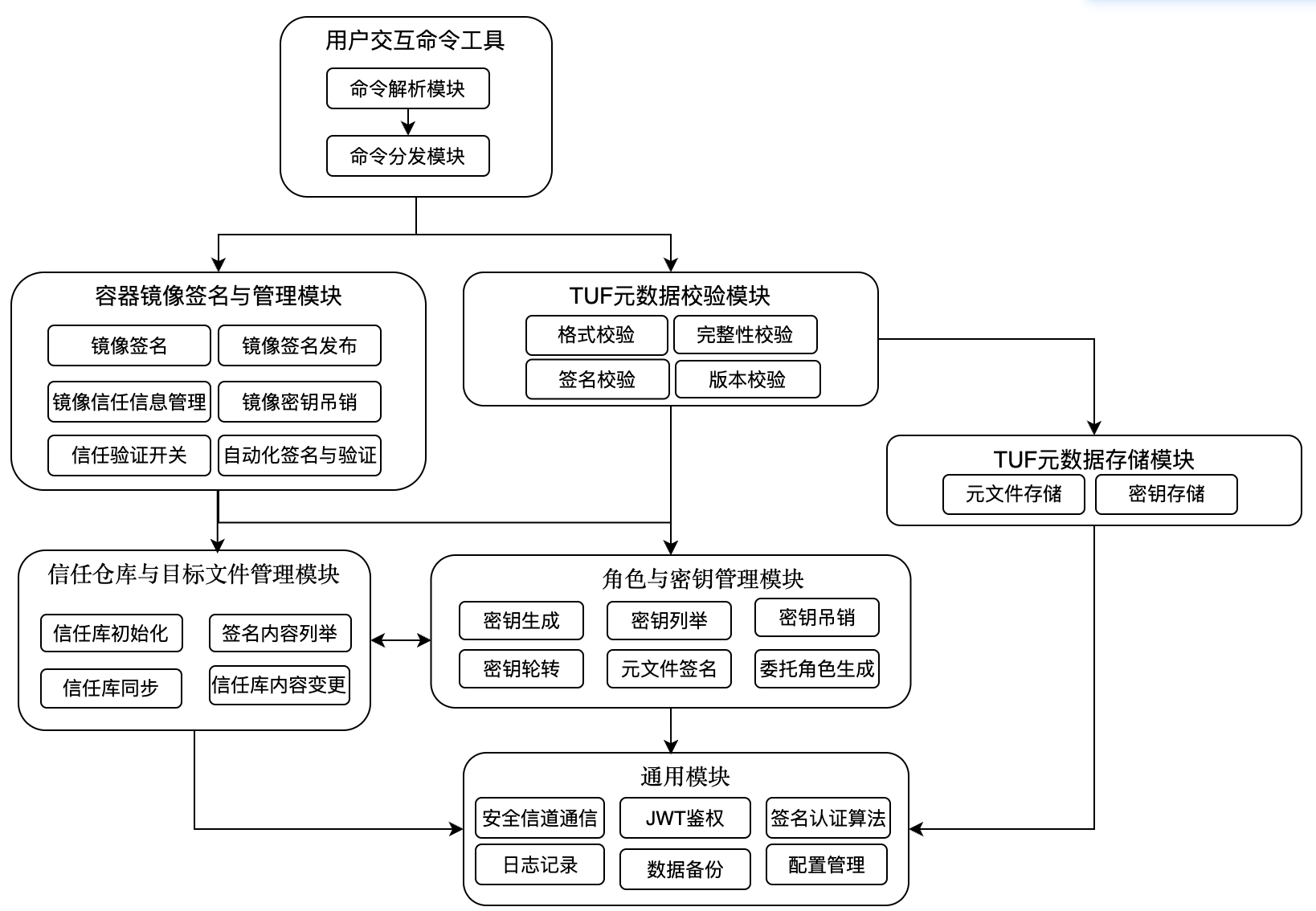
d. 调试和查看测试结果：如果测试用例失败，利用Google Test提供的详细失败信息，包括失败的测试用例、断言失败的具体行号、以及失败的断言类型。根据这些信息，可以定位并修复代码中的问题。使用调试工具（如 gdb 或 LLDB）来逐步执行测试用例，查看失败的根本原因。

## 集成测试策略

（1）确定参与集成测试的软件单元模块

镜像签名认证软件包含的关键软件单元包括容器镜像签名与管理模块、角色与密钥管理模块、信任(仓)库与目标文件管理模块、TUF元数据管理模块、通用功能模块以及辅助管理模块。

（2）确定不同软件模块之间的依赖关系，然后依据依赖关系由下层到上层的顺序逐层组装各个单元模块。



编写测试用例，验证由依赖关系的软件模块能够正确组装和交互。

（3）当所有模块组装到一起之后，通过设计典型的应用流程测试系统中多个模块之间可以正确协作完成系统级功能。

## 系统测试策略

* + 1. **功能测试方法**

在功能测试用例设计上遵循以下设计原则：

（1）采用了黑盒测试法，只通过软件的输入和输出验证软件的功能实现是否符合js协议的要求。

（2）从用户角度，逐个测试容器管理软件的内容信任相关的各个命令，测试关键功能

（3）分别在天脉容器、天脉Linux环境下进行兼容性测试。

* + 1. **性能测试方法**

（1）通过脚本连续创建不少于1000个软仓库公证，然后再用脚本循环对这些软仓库公证进行签名、拉取，发布等操作，通过行为观察判断是否发布成功。

（2）利用Jemeter等并发测试工具测试镜像签名认证软件服务器至少可支持100个客户端的并发访问。

（3）使用测试工具测试时间戳元数据文件以及快照元数据文件的生成和签名的延迟。

# 项目验收与交付

项目交付计划以下具体如1所示。

**表 1 交付计划说明**

|  |  |
| --- | --- |
| 交付物 | 说明 |
| 技术方案 | 2025年4月 |
| 需求文档 | 2025年6月 |
| 设计文档 | 2025年6月 |
| 源码 | 2025年8月 |
| 测试文档 | 2025年9月 |
| 验收文档：  软件用户手册  软件研制总结报告  软件溯源分析报告  软件知识产权说明 | 2025年10月前完成 |