专利申请技术交底书

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 交底书撰稿人 | 魏帅 | | | |
| 发明人 | 魏帅 | | | |
| 联系方式 | 电话 | 13161365651 | 邮箱 | 601770050@qq.com |
| **发明名称**  注：从技术方案的核心点出发取名 | 一种基于 RSA 加密算法和 JWT 的接口鉴权机制 | | | |
| **解决的技术问题**  注：可以是现有技术的缺陷，或者我们技术方案的带来的有益效果 | 本系统解决了对外开放API接口时的安全认证难题。具体解决了以下关键技术问题：  数据传输安全：通过RSA非对称加密确保敏感数据在传输过程中的安全性，避免数据被窃取或篡改。  身份认证：采用JWT（JSON Web Token）实现了可靠的身份认证机制，确保只有合法的第三方厂商能够访问API。  防重放攻击：通过Redis存储已使用的token并设置过期时间，有效防止请求被恶意重放。  灵活的权限控制：基于Spring AOP实现的注解式鉴权机制，使得权限控制既灵活又易于维护。  统一的异常处理：提供了规范化的异常处理机制，确保系统在遇到非法访问时能够返回统一的错误响应。  这些解决方案共同构建了一个安全、可靠、易于集成的API鉴权体系。 | | | |
| **技术方案的核心点**注：可以是我们技术方案跟现有技术方案的区别点；可以是我们技术方案的出彩点 | 本方案设计了一种创新的API鉴权机制，通过多层次的安全防护和灵活的控制策略，有效解决了对外开放接口的安全认证问题。其核心技术特点如下：   1. 创新的双层加密机制。首创性地将RSA非对称加密与JWT token相结合；外层RSA加密确保传输安全，内层JWT保证数据完整性和不可篡改；支持密钥定期轮换，显著降低密钥泄露风险。 2. 高效的防重放设计。基于JWT的issued at时间戳作为唯一标识符；结合Redis实现严格的一次性Token使用机制；支持分布式环境下的Token状态同步；相比传统时间窗口方案，完全杜绝重放攻击风险。 3. 动态的权限控制。基于Spring AOP实现注解式鉴权；支持方法级别的细粒度权限控制；可自定义校验规则和失败处理策略；业务代码与认证逻辑完全解耦。 4. 灵活的时效管理。创新地支持调用方通过请求头动态指定Token有效期；实现基于业务场景的精确超时控制；提供合理的有效期范围限制，避免滥用。 5. 完善的异常处理。支持自定义异常类型和错误消息模板；提供详细的错误原因定位；规范化的错误响应格式；完整的日志追踪机制。   这套方案通过创新的技术组合和精心的设计细节，在确保安全性的同时保持了较高的易用性和可维护性，特别适合需要对外开放API的企业级应用场景。 | | | |
| **技术领域**  注：应用的技术领域 | 本技术方案属于计算机网络安全技术领域，特别是Web应用安全和API访问控制领域。主要应用于企业级分布式系统的接口安全防护，涉及密码学、身份认证、访问控制等核心安全技术。该方案尤其适用于需要对外提供安全可靠的API接口的场景，如企业间系统集成、第三方应用接入等领域。 | | | |
| **术语解释**  注：对行业内的一些专用术语、专用英文缩略语进行英文全文、中文全文解释 | 1. RSA (Rivest-Shamir-Adleman)：英文全称：Rivest-Shamir-Adleman cryptosystem   ；中文解释：一种非对称加密算法，由Ron Rivest、Adi Shamir和Leonard Adleman在1977年提出；用途：用于数据加密和数字签名的公钥密码体制。   1. JWT (JSON Web Token)：英文全称：JSON Web Token；中文解释：基于JSON的开放标准（RFC 7519），用于在网络应用环境间传递声明；用途：用于身份验证和信息交换的紧凑且自包含的安全令牌。 2. AOP (Aspect-Oriented Programming)；英文全称：Aspect-Oriented Programming；中文解释：面向切面编程；用途：通过预编译和运行期动态代理实现程序功能的统一维护。 3. API (Application Programming Interface)：英文全称：Application Programming Interface；中文解释：应用程序编程接口；用途：定义应用程序之间进行信息交换的方法和数据格式。 4. Redis (Remote Dictionary Server)：英文全称：Remote Dictionary Server；中文解释：远程字典服务器；用途：开源的高性能键值对数据库，用于数据缓存和存储。 5. Token：中文解释：令牌；用途：用于身份验证和授权的一串字符串。 6. Header (HTTP Header)：中文解释：HTTP请求/响应头；用途：包含有关HTTP事务的操作参数，如身份验证信息等。 7. Timeout：中文解释：超时时间；用途：指定操作或令牌的有效期限。 8. iat (Issued At)：英文全称：Issued At Time；中文解释：签发时间；用途：JWT中表示令牌的签发时间的声明。 9. Authentication：中文解释：身份认证；用途：验证用户或系统身份的过程。 10. Authorization：中文解释：授权；用途：确定用户或系统访问权限的过程。 11. Replay Attack：中文解释：重放攻击；用途：描述攻击者通过重复发送之前捕获的有效数据包进行的攻击方式。 | | | |
| **背景技术介绍**  注：四段论  （可仿照百度百科里面对一些技术的介绍） | 1. 第一段：定义与发展。API接口鉴权是Web应用安全领域的重要组成部分，它确保只有经过授权的客户端才能访问服务器提供的接口。随着微服务架构和开放平台的普及，接口安全问题日益突出，传统的单一认证方式已难以满足现代企业级应用的安全需求。 2. 第二段：现有技术。目前主流的API鉴权方案包括Basic认证、OAuth2.0和JWT等。Basic认证方式简单但安全性较低；OAuth2.0框架功能完善但实现复杂；JWT提供了自包含的信息验证机制但存在token泄露风险。这些方案各有优劣，在实际应用中往往需要根据具体场景进行取舍。 3. 第三段：技术痛点。在实际应用中，企业对外开放接口面临着多重挑战：一是数据传输过程中的安全性问题；二是防范重放攻击的需求；三是不同业务场景下对认证有效期的差异化需求；四是系统性能与安全性的平衡问题。这些挑战需要一个更完善的技术方案来解决。 4. 第四段：发展趋势。随着安全威胁的不断演化，API鉴权技术正向着多层次防护、动态权限控制、智能风控等方向发展。将多种安全技术有机结合，构建完整的安全防护体系，已成为行业发展的主要趋势。在此背景下，设计一套兼具安全性、灵活性和易用性的API鉴权方案具有重要的现实意义。 | | | |
| **发明内容**  注：描述一下解决的技术问题，以及从方案的完整性角度来看，必须有哪些技术点，才能达到解决这个技术问题的目的。 | 一、技术问题本发明旨在解决企业对外开放API接口时面临的以下核心技术问题：  1. 数据传输安全问题：如何确保敏感数据在传输过程中不被窃取或篡改。  2. 身份认证可靠性问题：如何准确识别和验证第三方调用者的身份。  3. 重放攻击防护问题：如何有效防止黑客通过重放已捕获的请求进行攻击。  4. 认证时效性问题：如何根据不同业务场景灵活控制接口调用的有效期。  5. 系统性能问题：如何在确保安全的同时保持系统的高性能和可扩展性。  二、技术方案要点为全面解决上述技术问题，本发明必须实现以下关键技术点：  1. 双层加密体系。RSA非对称加密机制的实现；JWT token的生成与验证；密钥管理和定期轮换机制；加密数据的序列化和反序列化处理  2. 身份认证机制。基于注解的声明式认证；细粒度的权限控制；统一的身份验证切面；可扩展的认证规则引擎  3. 防重放保护。基于时间戳的请求唯一性校验；Redis分布式token存储；失效token的清理机制；并发请求的处理策略  4. 动态时效控制。可配置的token有效期；基于业务场景的超时策略；时效性验证的实现；过期处理机制。  5. 异常处理与监控。统一的异常处理框架；详细的错误信息追踪；系统运行状态监控。安全事件日志记录  6. 性能优化措施。验证逻辑的异步处理；缓存机制的合理使用；密钥操作的性能优化；分布式环境下的性能保证  通过以上技术点的有机结合，本发明构建了一个完整的API接口鉴权解决方案。该方案不仅解决了各项技术问题，还具备了良好的可扩展性和可维护性。方案的实现既保证了接口调用的安全性，又兼顾了系统的性能要求，为企业对外开放API提供了可靠的安全保障。 | | | |
| **具体实施示 例**  注：从方案中实现同样目的技术手段可能有很多种，这样可以要求比较大的专利保护范围。产品的话，可以结合产品的线路构造图进行说明。方法的话，可以结合流程处理图进行说明。 | 1. 整体架构实现   本发明提供了一种基于RSA加密算法的API接口鉴权机制，主要包含以下实施部分：   1. 密钥管理实施   public class RSAUtils {  // RSA密钥对生成  public static RSAKeyPair generateKeyPair() {  KeyPair pair = SecureUtil.generateKeyPair("RSA");  // 转换为Base64编码  String pubEncBase64 = Base64.encode(pair.getPublic().getEncoded());  String priEncBase64 = Base64.encode(pair.getPrivate().getEncoded());  return new RSAKeyPair(pubEncBase64, priEncBase64);  }  }   1. Token生成与加密实施   public class JWTUtils {  public static String createToken(Map<String, Object> payload, int timeout, String key) {  DateTime now = DateTime.now();  DateTime expTime = now.offsetNew(DateField.SECOND, timeout);  // 设置JWT标准声明  payload.put(RegisteredPayload.ISSUED\_AT, now);  payload.put(RegisteredPayload.EXPIRES\_AT, expTime);  payload.put(RegisteredPayload.NOT\_BEFORE, now);  return JWTUtil.createToken(payload, key.getBytes());  }  }   1. 注解式权限控制实施   @Target(ElementType.METHOD)  @Retention(RetentionPolicy.RUNTIME)  @Documented  public @interface HeaderAuth {  String headerTokenKey() default "X-Access-Token";  boolean requiredToken() default true;  String headerTimeoutKey() default "timeout";  String expectedValue() default "";  Class<? extends RuntimeException> RuntimeException() default ForbiddenException.class;  String message() default "Header验证失败";  }  二、鉴权流程实施  1. 请求预处理阶段。获取请求头中的加密token和超时设置；验证参数完整性和有效性；提取必要的认证信息  2. 解密验证阶段。使用RSA私钥解密token；验证JWT签名和有效期；检查token是否重复使用  3. 权限控制阶段。解析JWT中的权限信息；验证访问权限；记录访问日志  4. 异常处理阶段。捕获并处理验证异常；生成标准错误响应；记录安全事件日志  三、防重放实施  private void matchValue(HeaderAuth auth, String actualTokenValue, int timeoutValue) {  // 解密token  String decryptContent = RSAUtils.decrypt(actualTokenValue, privateKey);    // 获取签发时间作为唯一标识  int iat = JWTUtils.getPayload(decryptContent, RegisteredPayload.ISSUED\_AT);  String timestamp = String.valueOf(iat);    // 检查是否已使用  String tokenValue = RedisUtil.get(timestamp);  if (StringUtils.isNotBlank(tokenValue)) {  throw createException(auth, "Token已使用");  }    // 存储token并设置过期时间  RedisUtil.set(timestamp, actualTokenValue, timeoutValue);  }  四、流程图 | | | |
| **权利要求**  注：罗列一下，我们的技术方案里面，最想保护的技术点 | 权利要求1（主权利要求）：一种基于RSA加密算法的API接口鉴权方法，其特征在于，包括以下步骤：接收包含加密token和超时时间的API请求；使用RSA私钥解密token获取JWT内容；基于时间戳进行一次性token验证；验证通过后执行业务逻辑。  权利要求2：根据权利要求1所述的鉴权方法，其特征在于，所述一次性token验证包括：从JWT中提取签发时间作为唯一标识；使用分布式缓存检查该标识是否已被使用；若未使用，则存入缓存并设置过期时间；若已使用，则判定为重放攻击并拒绝请求。  权利要求3：根据权利要求1所述的鉴权方法，其特征在于，所述超时时间通过以下方式控制：在请求头中指定timeout参数；验证timeout值是否在合法范围内；将timeout值用于设置token的缓存过期时间；超过时间后自动失效。  权利要求4：根据权利要求1所述的鉴权方法，其特征在于，包括基于注解的权限控制：通过注解声明需要鉴权的接口；支持自定义请求头参数名；支持自定义异常处理；支持细粒度的接口权限控制。  权利要求5：根据权利要求1所述的鉴权方法，其特征在于，JWT的处理包括：验证签名的有效性；检查token的生效时间；检查token的过期时间；验证token的完整性。  权利要求6：根据权利要求1所述的鉴权方法，其特征在于，异常处理包括：记录详细的错误原因；返回标准化的错误响应；支持自定义异常类型；提供完整的异常追踪信息。  权利要求7：根据权利要求1所述的鉴权方法，其特征在于，性能优化措施包括：使用分布式缓存存储token状态；异步处理日志记录；优化RSA解密操作；支持分布式环境部署。  这些权利要求涵盖了本技术方案的核心创新点，包括：双层加密保护机制；基于时间戳的防重放设计；动态超时控制；注解式权限控制；分布式token管理；完整的异常处理体系。 | | | |
|  |  | | | |