移动云安全支付系统设计与实现

曹鹏飞 李 杰 杨 君*
(三江学院计算机科学与工程学院 江苏 南京 210012)

摘 要 以第三方支付平台为移动云支付系统基础,从支付方式、支付风险、认证鉴权、加密方式、通信安全等方面对现有的移动支付系统进行研究。针对普通静态二维码易伪造、易被篡改、难辨别等问题,设计一套基于TOTP算法的动态二维码移动云支付方案。采用基于朴素贝叶斯算法和自定规则进行风险评估,经过测试能够在一定程度上提升移动支付系统的安全性。

关键词 移动支付 云支付 支付系统 安全支付

中图分类号 TP3 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2019.04.047

DESIGN AND IMPLEMENTATION OF MOBILE CLOUD SECURITY PAYMENT SYSTEM

Cao Pengfei Li Jie Yang Jun*
(*College of Computer Science and Engineering, Sanjiang University, Nanjing 210012, Jiangsu, China*)

Abstract Based on the third-party payment platform, this paper studied the existing mobile payment system from the aspects of payment mode, payment risk, authentication, encryption mode and communication security. Aiming at the problems of forgery, tampering and discrimination of ordinary static QR code, we designed a mobile cloud payment with a dynamic QR code based on TOTP algorithm. Risk assessment was based on naive Bayesian algorithm and self-defined rules. After testing, this method can improve the security of mobile payment system to a certain extent.

Keywords Mobile payment Cloud payment Payment system Security payment

0 引 言

现如今移动支付已成为人们日常生活中不可缺失的一部分,用户使用智能手机就可以完成支付业务,技术主要有:蓝牙支付技术、NFC 支付技术、二维码支付技术、人脸识别支付技术[1-2]。

随着移动支付技术的发展,移动支付的安全问题随之出现,众多用户使用静态二维码作为收款码,但静态条码容易被伪造,误导客户扫描伪码,实施欺诈,导致账户资金被盗刷、个人敏感信息泄露等风险问题。在移动支付交易过程中,移动终端与支付平台之间的通信安全也至关重要,一旦交易过程中的数据被窃取,信息被篡改。不法分子就能够获得用户、商户交易的相关信息,并对交易过程的数据进行伪造、重放等攻击。

本文针对移动支付中二维码易伪造、易被篡改、变造,易携带木马等问题,设计一套基于TOTP算法的动态二维码支付方案,保证交易的不可为造性、实时性、唯一性、时效性、可认证性、完整性。

1 方案设计

1.1 总体架构设计

系统由多终端设备与云支付平台组成,各终端通过设计的REST接口与云支付平台进行交互。系统共分为三层:访问层、服务层、资源层。总体系统框架如图1所示。

收稿日期:2018-10-09。江苏省教育厅自然基金项目(17KJD520007)。曹鹏飞,讲师,主研领域:网络安全。李杰,实验师。杨君,副教授。

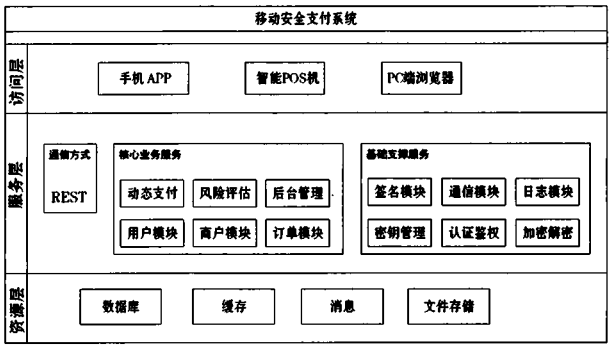

图 1 移动安全支付系统总体架构

访问层即访问云端的方式，分别有手机 APP、智能 POS 机、PC 浏览器。

服务层是整个移动支付系统的核心，包含了云端支付系统的功能，包括：核心业务服务和基础支撑服务。核心业务服务包含动态支付、风险评估、后台管理、用户模块、商户模块、订单模块。基础支撑服务包含签名、通信、日志、密钥管理、认证鉴权、加密解密等模块。

资源层主要是存放移动支付系统信息的实体，使用关系型数据库存储账户信息、支付信息、商户信息、商品信息等支付平台数据。使用分布式缓存（Redis）对热点数据进行缓存。随着系统业务量增大，采用 RabbitMQ 作为消息系统，能够解决数据投递，非阻塞操作和推送通知，异步处理和工作队列的问题等。针对一般文件存储，可使用第三方存储服务，利用主流 CDN 和自建 CDN 节点的优势，在性能和成本之间寻求平衡。

1.2 主要功能设计

1.2.1 动态支付功能设计

动态支付是移动云支付系统的核心部分，支付密钥是标记交易的唯一标识符，只有商家通过发送订单信息和客户生成的支付密钥才能申请交易。云端对支付密钥进行认证后完成支付交易，支付流程如图 2 所示。

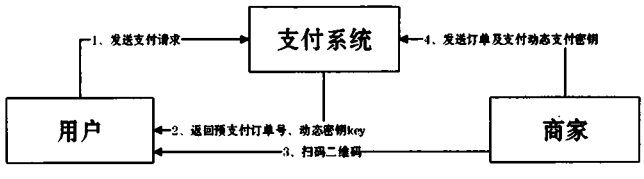

图 2 支付流程

支付流程由两个部分构成：

1）支付二维码产生 用户在支付时产生的二维码由基于时间规则算法动态生成的。在整个支付过程中支付二维码每 60 s 生成一次，每个用户的支付二维码都是唯一的。二维码有效时间在支付完成或超时立即失效。相对于静态二维码的易伪造、成本低、易窃取等缺点，动态二维码能保证其不可为造性、实时性、唯一性、时效性、可认证性、完整性。

2）支付密钥认证 云移动支付系统收到交易订单和支付密钥后，要对密钥进行基于时间的认证。先验证是否在有效期内，然后通过 TOTP 算法[3]生成 6 位动态口令，将 6 位动态口令和时间戳合成的字符串通过哈希算法 SHA-256 生成动态支付密钥[3]。当云端服务器接收到交易过程产生的动态支付密钥后，根据支付订单号查询其所对应的 TOTP 中密钥串 K 和时间戳，与当前时间生成支付密钥并与用户生成的支付密钥进行验证。如相同则交易正确，否则表明支付超时或被篡改伪造。

1.2.2 风险评估功能设计

用户的风险评估是在商户发送收款请求时进行的。针对用户的个人信息、消费水平、消费时间、地点和习惯等方面进行安全性分析，评估当前消费是否为该用户的正常购物行为。风险评估流程如图 3 所示。

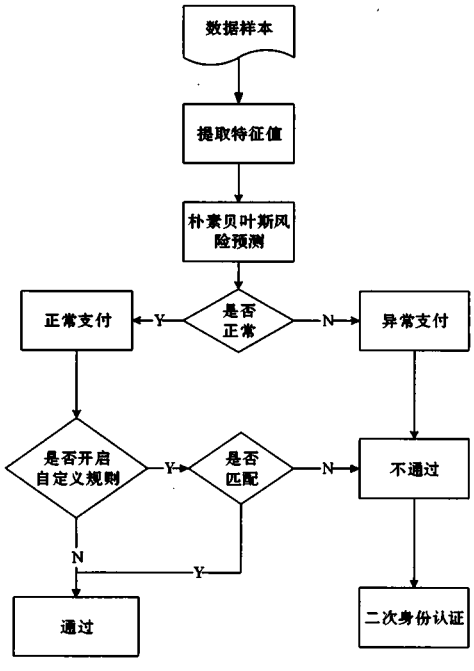

图 3 风险评估流程

系统采用朴素贝叶斯算法[4]与自定义规则匹配相结合的方式来实现支付风险评估。根据用户交易过程中产生的样本数据，提取特征值属性，对用户的支付行为行进分类。若结果异常，则交易不被允许，要求用户进行二次身份认证。若结果正常则对用户支付的相关数据进行规则匹配。自定义规则匹配用户常用的地点、支付时间、支付金额等限制。如果在用户预设范围内则通过则允许支付，否则要求二次身份认证。

1.2.3 认证鉴权模块设计

在移动云支付系统中认证鉴权包括：统一的认证

服务和统一鉴权服务。

统一认证服务主要是在用户执行操作时,提供我是我本人的身份证明[5]。常见的身份认证用户名和密码的输入是存在被窃取的风险。现如今人脸识别、指纹、短信、声纹是用户认证主要方式。能够有效地避免不法分子通过自动化工具进行密码爆破和窃取。

统一鉴权服务是在用户认证会话的维持和维护有效性[6]。大多数 Web 站点使用 Session,其主要功能是保持会话信息,对用户相关信息的存储和对用户进行验证,而在移动系统中要使用 Token 来对用户的身份鉴权。

认证鉴权主要流程如下:

1) 客户端通过获取登录验证码请求提交用户手机号获取登录验证码。

2) 云端支付平台生成 6 位随机验证码,通过第三方短信/语音平台发送至用户手机。

3) 客户端得到短信/语音验证码后进行登录请求。

4) 云端支付平台短信/语音验证通过后,云端支付平台对用户进行登录地点/时间异常检查。

5) 如果用户不在常用的地点或不在常规时间段登录,将会要求用户输入密码进行二次身份认证。

6) 云端支付平台验证后通过相应的方法生成一个 Token 与该用户进行关联,并生成的 Token 返回给客户端。

7) 客户端在所有的请求中都需要携带 Token 进行认证,云端支付平台通过解析 Token 来对客户端进行身份检测。

8) 当用户退出登录或在其他设备登录时,之前的 Token 将会失效。

1.2.4　通信功能模块设计

通信功能模块是连接移动端和云端支付平台的桥梁,用来保证用户、商户和云端支付系统之间信息传递的安全性。

为保证交易安全性,系统传输模式使用 HTTPS 协议,使用 JSON 格式作为数据的载体,对通信过程中关键数据进行加密操作。对 JSON 数据进行数字签名,能够有效地保证数据完整性、机密性和不可抵赖性。

在通信过程中采用 AES 加密数据,使用 RSA 密钥对 AES 共享密钥进行二次加密。云端支付系统为每一个用户生成私钥,而公钥存储在云端,具体加密方案设计流程如下:

1) 云端支付平台随机生成 16 个数字 Key。

2) 云端支付平台使用 RSA 公钥将随机生成的 Key 加密生成 AESkey。

3) 云端支付平台通过 AES 算法结合 Key 对关键数据加密。

4) 客户端通过提取 JOSN 数据中 AESkey,使用 RSA 私钥对 AESKey 解出 Key。

5) 客户端通过 AES 算法结合 Key 对关键数据解密。

2　系统实现

移动安全支付系统采用 Android 系统作为用户客户端,智能 POS 机作为商户客户端。整个云端支付系统使用 Redis 作为高速缓存,使用 RabbitMQ 作为消息队列系统,完成各个模块之间的通信,使用 Celery 作为任务分发模块。

2.1　动态支付模块实现

动态支付是整个移动支付平台中关键的功能。当用户在线下商店选完商品需要支付时,客户端通过使用 OKHttp 发送支付请求。当云端支付系统验证请求合法性后,通过从数据库设定的 Sequence 中取出序列号和当前日期组成唯一的预支付订单号和生成的 16 位随机密钥 Key、当前时间戳存入数据库并返回给用户。

如图 4 所示,客户端接收到云端支付系统返回的数据后,客户端利用本地共享密钥和随机密钥生成新的密钥 K,通过 TOTP 算法生成了 6 位动态口令。之后利用哈希函数 HASH-256 对 6 位动态口令与云端返回的时间戳拼接成的字符串进行生成不可逆的支付密钥。最后客户端利用 ZXing[8] 将“预支付订单号 & 支付密钥”拼接成的字符串生成支付二维码,支付二维码随着时间每 60 s 更换一次。

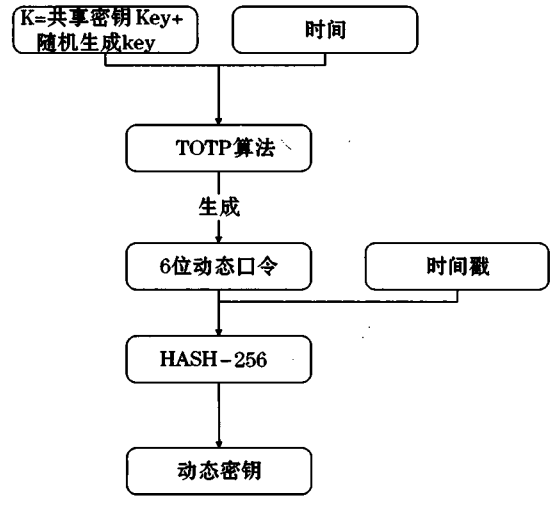

图 4　动态密钥生成流程

云端支付系统在收到订单信息后，通过查询预支付订单表，根据支付密钥规则重新生成当前密钥同订单中的商户上传的支付密钥进行比对。如果相同就通过事务处理扣除用户相应的支付金额，增加商户的资金。如果不相同，则返回支付失败，并记录。

2.2 风险评估模块实现

该模块由 SciKit learn 实现，它是一款数据挖掘和数据分析工具，拥有分类、回归、聚类、降维等相关学习算法和模型[9]。

本文通过提取用户支付地点、支付商户名、支付商品数量、支付时间和支付金额这五个维度特征数据进行训练，对系统支付过程的风险进行评估。

当商户发出交易指令后，系统自动提取支付地点、支付商户名、商品数量、支付时间和支付金额，使用朴素贝叶斯算法进行训练样本模型，最后对本次用户支付行为进行支付风险判断。

用户行为通过了朴素贝叶算法的异常识别后，系统将再进行规则匹配。如果匹配不通过，则要求用户输入支付密码，进一步确认身份，完成认证。

2.3 认证鉴权模块实现

如图 5 所示，系统生成 6 位随机数后通过 Celery 异步发送给用户，并设置过期时间存入缓存 Redis 中。

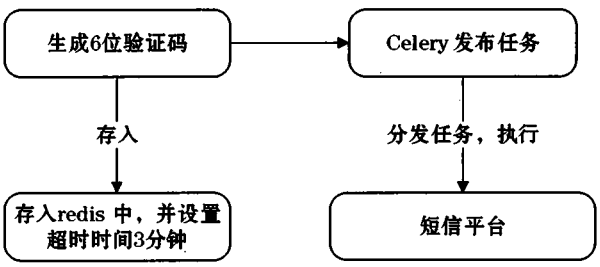

图 5 短信发送流程

返回的数据由三个属性构成：用户手机号、超时时间、验证码。将用户手机号码作为 Key，验证码作为 Value。云端支付系统直接从 Redis 中获取号当前手机对应的验证码与用户输入进行验证，验证通过将用户 ID 转换生成携带身份信息的令牌值（Token ID）。具体过程如下：

1）JWT 头部 header 中 typ 为 type 缩写，代表是 JWT，alg 表示加密方式为 HS256，exp 代表当前时间。

2）在 JWT 的消息体 playload 中添加 user_id 和时间戳。

3）签名（Sign）。其过程如图 6 所示，把 header、playload 分别使用 Base64 编码，使用‘.’将经过编码后的字符串进行连接，执行 HMAC-SHA-256 算法加密，对加密后的数据使用 Base64 编码，生成签名的字符串。

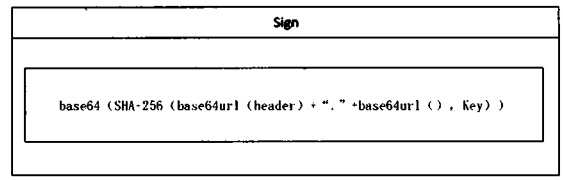

图 6 Sign 过程

4）生成 Token。把 Base64 编码后 header 和 playload 以及 sign 用‘.’连接起来即 header. playload. sign，就生成了 Token。

5）校验 Token。首先用将字符串按‘.’切分三段字符串，分别得到 header、playload 和 sign。然后将 header. playload 拼装用密钥和 HAMC SHA-256 算法进行加密然后得到新的字符串和 Sign 进行比对。如果一致数据正确，则从 head 取出 exp 对存活期进行判断，如果超时则要求重新登录，如果在存活期内返回 user_id 值。

用户短信登录认证成功后，系统将签发两个 Token：

1）Token，短期证书，用于客户端的正常服务请求所携带的认证信息。

2）Refresh Token，较长期的证书，用于 Token 失效后，自动申请新的 Token 所需携带的认证信息，不可直接用于服务请求。

分别给这两个 Token 设置不同的有效期。在用户退出登录时通过客户端将 Token 和 Refresh Token 进行销毁。在 Refresh 有效期内，在 Token 过期时客户端通过 Refresh Token 获取新的 Token。当 Refresh 过期后，需要用户重新认证登录。

3 核心算法

3.1 基于 TOTP 算法的动态支付密钥技术

TOTP（基于时间的一次性密码算法）是支持时间作为动态因素基于 HMAC 一次性密码算法的扩展[10]。

TOTP 计算公式如下：

$$TOTP(K,C) = Truncate(HMAC\text{-}SHA\text{-}1(K,C)) \quad (1)$$

式中：K 是密钥串（Base32 字符串）；C 是时间戳。

$$C = (T - T_0)/X \quad (2)$$

式中：T 是当前时间戳；T_0 取值为 0；X 表示时间步长，默认是 30 s。

HMAC-SHA-1 是从 SHA1 哈希函数构造的一种哈希算法，HMAC 将密钥与消息数据混合，输出长度为 160 位哈希。

Truncate 是函数，实现如下：

1）将密钥串 K 与时间戳 C 通过 HMAC-SHA-1 加密,得到一个长度为 20 字节的密串。

2）取出密串的最后一个字节的低 4 位,作为截取加密串的下标偏移量。

3）按照下标偏移量开始,获取 4 个字节,按照大端方式组合成一个整数。

4）截取这个整数的后 6 位转成字符串返回。

如图 7 所示,因为 TOTP 的动态性,其中密钥串 K 分别由静态密钥串 key1 和动态密钥串 key2 组成。静态密钥串 key1 是客户端与云端支付平台的共享密钥;动态密钥 key2 是由云端支付平台随机生成并下发给客户端。客户端则通过 TOTP 算法生成 6 位动态口令,将生成的动态口令与时间戳相结合,最后通过哈希算法 SHA-256 生成动态支付密钥。

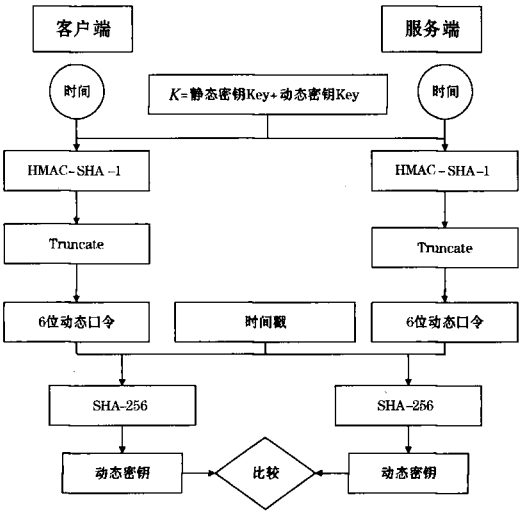

图 7　TOTP 算法的生成与认证

3.2　基于朴素贝叶斯算法的风险评估算法

朴素贝叶斯(Naive Bayes)是贝叶斯算法中的分支之一,是常见的一种分类方法,简称 NB 算法[11]。公式如下:

$$P(C|X)=\frac{P(X|C)P(C)}{P(X)} \tag{3}$$

式中:C 为类别;X 为待分类项;$P(C|X)$ 为 X 的状态下是 C 的概率大小。求解过程如下:

1）设置待分类项 $X=\{a_1,a_2,\cdots,a_m\}$,每个 a 为 X 一个特征属性。

2）确定类别集合 $C=\{Y_1,Y_2,\cdots,Y_m\}$。

3）计算 $P(Y_1|X),P(Y_2|X),\cdots,P(Y_n|X)$。

4）如果 $P(Y_k|X)=Max\{P(Y_1|X),P(Y_2|X),\cdots,P(Y_n|X)\}$,则 X 属于 Y_k 类。

如图 8 所示,系统选取支付时间,支付地点、商户、支付金额、支付商品数量五个维度作为特征属性,设定用户的支付地点为 L,支付时间 T,支付商户 S,支付商品 G,支付金额 M,即 $X=\{L,T,S,G,M\}$。

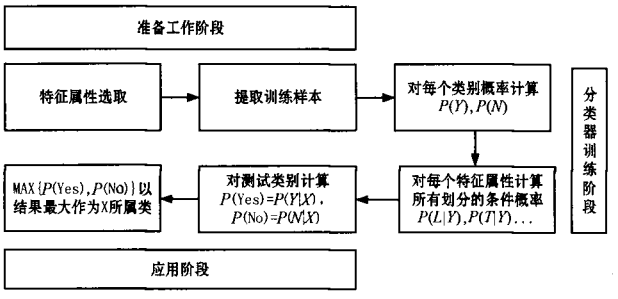

图 8　贝叶斯风险评估流程图

将 X 代入式(3)运算:

$$P(Y|X\{L,T,S,G,M\})=\frac{P(X|T)\times P(Y)}{P(X)} \tag{4}$$

结果如下:

$$\frac{P(Y)\times P(X|Y)}{P(X)}=$$

$$\frac{P(Y)\times P(L|Y)P(T|Y)P(S|Y)P(G|Y)P(M|Y)}{P(L,T,S,G,M)} \tag{5}$$

式中:$P(L|Y)$ 正常支付情况下 L 的概率;$P(L)$ 为所有情况下 L 的概率。计算 $P(Y|X)$ 支付正常为 $P(\text{Yes})$,计算 $P(N|X)$ 支付异常为 $P(\text{No})$,如 $P(\text{Yes})>P(\text{No})$ 则表明支付正常,反之则支付异常。

4　系统运行与测试

4.1　系统运行

本系统中所需要的服务器采用腾讯云 CVM 服务器,整体部署方案如图 9 所示。整个云端支付系统使用 Nginx 作为 HTTPS 服务器和反向代理,不仅可以提高 Python 的 I/O 处理能力,还可以缓冲响应,减轻后端压力。Gunicorn 作为 Python 应用服务器,易于配置管理,能够按需求切换异步和同步工作模式。数据库服务器采用 MySQL,缓存服务器使用 Redis,消息队列使用 RabbitMQ。Celery 作为分布式任务队列系统。

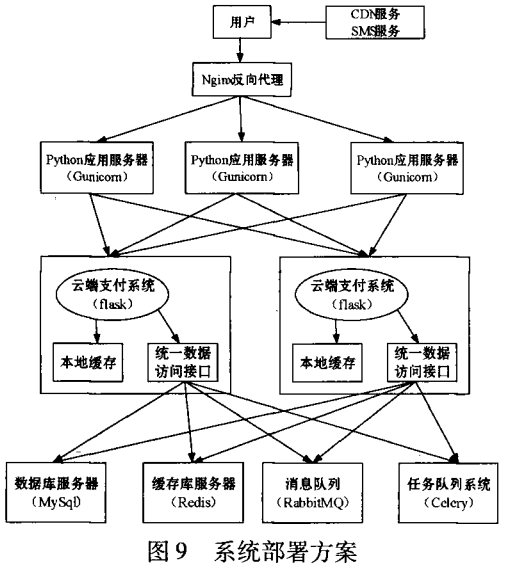

图 9　系统部署方案

1）服务器端软硬件配置环境：

（1）服务器类别：腾讯云 CVM 服务器。

（2）服务器机型配置：标准型 S1；CPU：16 核；内存：8 GB；硬盘：带宽：100 Mbit/s。

（3）操作系统：Ubuntu Server 16.04.1 LTS 64 位。

（4）应用服务器：Nginx 1.10.3（Ubuntu），Gunicorn-19.8.1；Python 3.6。

（5）数据库系统：MySQL 5.6，Redis 3.2.9。

（6）消息队列系统：RabbitMQ 3.5.7-1。

（7）分布式任务队列：Celery 4.1.0。

2）客户端软硬件配置环境

客户端使用安卓（Android）系统作为运行终端。运行配置如下：

（1）手机型号：ONEPLUS A3010；

（2）安卓版本：Android 7.1.1；

（3）运行内存：6 GB；

（4）存储内存：128 GB。

云端系统搭建完成后，在两部手机上安装移动支付 APP。一部充当普通用户，另一部当作商户，分别注册各自账号。用户在需要购买商品时，在商家确定商品后，用户只在需要支付的时候打开支付二维码，商家进行扫描二维码收款。其中二维码每 60 s 更新一次，动态更新达到了基本的设计需求。系统运行如图 10 所示。

图 10 系统运行

4.2 系统测试

4.2.1 测试指标

本系统测试分为风险评估测试、主要功能测试与性能测试。

风险评估测试利用朴素贝叶斯算法在实验环境进行训练，对用户的异常支付行为进行识别和匹配。

功能测试主要使用黑盒测试[12]，通过各个功能编写测试用例，按照测试用例进行测试，验证结果的正确性。检查系统各个功能是否能够满足生成环境需求，包括短信登录认证、密钥生成分发、机密信息加密解密、动态支付密钥生成与认证、风险评估等功能模块。

性能测试通过自动化的测试工具来对系统的各项性能指标进行测试[13]。系统性能测试主要测试系统的高并发性能，包括：平均响应时间、每秒处理请求数量、平均流量和错误率等。

4.2.2 风险评估测试

在实验环境下使用朴素贝叶斯算法进行训练，通过统计两个用户的购物数据，分别训练或测试使用的白样本和黑样本。通过人工标签 Label（0 为正常，1 为异常）共 600 条，其中 150 份为异常样本，450 份为正常样本，部分训练样本如表 1 所示。

表 1 训练样本

Time	Location	Store	Money	Number	Label
15	3010	12	3	1	0
11	5210	254	45	1	0
18	3056	111	86	2	0
0	3010	225	60	3	1
9	3010	125	100	4	1
20	5210	21	128	2	0
13	3010	52	6	1	1
15	5210	111	50	5	1

通过 Scikit-Learn 中的 GaussianNB 函数进行训练，采用拉普拉斯平滑法，对每个分量 x 的计数加 1 可以避免零概率问题。

实验将数据样本分为 3 份，采用 3 折交叉验证，两份为训练集，另一份为测试集。具体异常判断结果统计如表 2 所示。

表 2 判断结果统计表

标签名	正常支付	异常支付
判断属于正常支付	121	12
判断属于异常支付	29	38
合计	150	50

对样本数据进行打乱,再次进行三次 3 折交叉验证。具体数据如表 3 所示。

表 3　三次 3 折交叉验证统计表

3 折交叉验证	正确率	召回率	综合评价指标
第一次	0.851	0.82	0.835
第二次	0.795	0.88	0.835
第三次	0.864	0.84	0.851
均值	0.836	0.846	0.840

经测平均准确率能达到 80% 以上,召回率在 84% 左右,综合标价指标均达 0.8,而且数据都比较平稳,没有太大的浮动。可以看出朴素贝叶斯算法在一定程度上能有效地辨别异常支付的能力,再结合移动支付系统的自定义匹配规则,能够对大部分的危险支付行为进行预警,有效地保障用户的资金安全。

4.2.3　主要功能测试

Token 获取、密钥更新、生成测试、风险评估功能测试,如表 4 所示。

表 4　功能测试用例

测试内容	Token 测试	密钥更新、生成测试	风险评估功能测试
测试目的	验证 Token 是否正常	验证密钥对中私钥是否正常生成、更新	验证能否识别风险。
测试方法	(1)输入手机号码,获取验证码 (2)验证码发送给云端支付系统 (3)云端支付返回 Token Refresh Token	(1)云端支付系统发送支付请求、获取验证码 (2)客户端解析动态口令和时间戳 (3)客户端生成支付密钥 (3)提交给云端支付系统验证 (4)验证通过返回私钥	(1)用户在同一城市多次支付 (2)修改地点属性执行支付操作 (3)云端支付系统风险评估
预期结果	返回 Token Refresh Token	返回私钥 动态密钥生成成功	需二次验证
结果	通过	通过	通过

4.2.4　性能测试

本系统性能测试主要测试系统的高并发性能,包括:平均响应时间、每秒处理请求数量、平均流量和错误率等。具体测试用例如表 5 所示。采用专业测试工具 Apache Jmeter[14],用于压力和性能测试,测试结果如表 6 所示。

表 5　性能测试用例

测试内容	高并发性能测试
测试目的	验证系统在高并发环境下是否能正常工作
测试方法	(1)运行 Jmeter,选择测试端口 443,协议 Https (2)在 SSL 管理器中选择证书 CA (3)添加线程组,配置相关参数 (4)开始测试
预期结果	(1)服务器运行正常 (2)服务器能承受 400QPS 访问量
结果	(1)服务器运行正常,符合预期 (2)实测服务器能承受 501QPS 访问量

表 6　性能测试表

测试数量/个	并发线程/个	平均响应时间/s	每秒请求数量/个	平均流量/(KB·s^{-1})	错误率/%
25 781	200	0.234	501	500	0

根据测试数据看出系统整体性能良好,在网络状态良好的情况下延迟较低,响应及时,能正常运行。

5　结　语

本文为移动云安全支付设计了一套可靠的解决方案。通过与 TOTP 算法相结合设计出一种新型动态二维码支付方式,保证二维码的不可伪造性、唯一性、时效性等安全性;通过改进现有的 JWT 协议建立完整的认证鉴权模式;在通信过程中,采用数字签名确保数据的完整性、安全性,同时具有不可抵赖性和防止重放等功能;采用了一种基于朴素贝叶斯算法和自定规则的风险评估模型,经过测试能够在一定程度上提升移动支付系统的安全性。

参　考　文　献

[1] 陈慧慧. 新型移动支付创新发展及安全监管研究[J]. 电信网技术, 2016 (5):37 - 39.

[2] Lu J, Yang Z, Li L, et al. Multiple schemes for mobile payment authentication using QR code and visual cryptography [J]. Mobile Information Systems, 2017, 2017, Article ID 4356038:1 - 12.

[3] Burch L L, Buss D F, Henderson L H. Time-based one time password (totp) for network authentication: U. S. Patent Application 15/138,521[P]. 2016:8 - 18.

[4] 叶云,石聪聪,余勇,等. 保护隐私的分布式朴素贝叶斯挖掘[J]. 应用科学学报, 2017,35(1):1 - 10.

[5] 张建标,徐万山,刘国杰,等. 一种结合网络行为分析的可信连接架构[J]. 信息网络安全, 2018(3):78 - 85.

(下转第 322 页)

[9] Otmani A, Tillich J P, Léonard Dallot. Cryptanalysis of Two McEliece Cryptosystems Based on Quasi-Cyclic Codes[J]. Mathematics in Computer Science, 2010, 3(2):129 - 140.

[10] Faugère J C, Otmani A, Perret L, et al. Algebraic Cryptanalysis of McEliece Variants with Compact Keys[C]//Proceedings of the 29th Annual international conference on Theory and Applications of Cryptographic Techniques. Berlin: Springer-Verlag, 2010: 279 - 298.

[11] Guo Q, Johansson T, Stankovski P. A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors[C]// International Conference on the Theory and Application of Cryptology and Information Security—ASIACRYPT 2016. Springer Berlin Heidelberg, 2016: 789 - 815.

[12] Janwa H, Moreno O. McEliece Public Key Cryptosystems Using Algebraic-Geometric Codes [C]//IEEE International Symposium on Information Theory. IEEE, 1996.

[13] Couvreur A, Marquez-Corbella I, Pellikaan R. Cryptanalysis of mceliece cryptosystem based on algebraic geometry codes and their subcodes[J]. IEEE Transactions on Information Theory, 2017, 63(8):5404 - 5418.

[14] Goppa V D. Codes on algebraic curves[J]. Soviet Mathematics Doklady, 1981, 24:170 - 172.

[15] Fulton W. Algebraic curves—an introduction to algebraic geometry(reprint vrom 1969)[M]. Boston: Addison-Wesley, 1989.

[16] Alzubi J, Alzubi O, Chen T. Forward error correction based on algebraic-geometric theory[M]. Berlin: Springer, 2014.

[17] Justesen J, Larsen K J, Jensen H E, et al. Construction and decoding of a class of algebraic geometry codes[J]. IEEE Transactions on Information Theory, 1989, 35(4):811 - 821.

[18] Silverman J H. The arithmetic of elliptic curves[M]. Berlin: Springer, 2009.

[19] Schoof R. Nonsingular plane cubic curves over finite fields [J]. Journal of Combinatorial Theory, Series A, 1987, 46 (2):183 - 211.

[20] Prange E. The use of information sets in decoding cyclic codes[J]. IRE Transactions on Information Theory, 1962, 8 (5):5 - 9.

[21] Lee P J, Brickell E F. An observation on the security of mceliece's public-key cryptosystem[C]//Advances in Cryptology — EUROCRYPT'88: Workshop on the Theory and Application of Cryptographic Techniques. 1988: 275 - 280.

[22] Stern J. A method for finding codewords of small weight[C]// International Colloquium on Coding Theory and Applications. Springer, Berlin, Heidelberg, 1988.

[23] Becker A, Joux A, May A, et al. Decoding Random Binary Linear Codes in $2^{n/20}$: How 1 + 1 = 0 Improves Information Set Decoding[C]//International Conference on Theory & Applications of Cryptographic Techniques. Springer-Verlag, 2012.

[24] May A, Ozerov I. On Computing Nearest Neighbors with Applications to Decoding of Binary Linear Codes[M]//Advances in Cryptology—EUROCRYPT 2015. Springer Berlin Heidelberg, 2015.

[25] Both L, May A. Decoding Linear Codes with High Error Rate and Its Impact for LPN Security[C]//International Conference on Post-quantum Cryptography. Springer, Cham, 2018.

[26] Berson T A. Failure of the mceliece public-key cryptosystem under message-resend and related-message attack[C]//Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology. London: Springer-Verlag, 1997: 213 - 220.

[27] Kobara K, Imai H. Semantically secure mceliece public-key cryptosystems-conversions for mceliece PKC [C]//Public Key Cryptography. 2001.

[28] Kiltz E, Mohassel P, O'Neill A. Adaptive trapdoor functions and chosen-ciphertext security[C]//Proceedings of the 29th Annual international conference on Theory and Applications of Cryptographic Techniques. Berlin: Springer-Verlag, 2010: 673 - 692.

[29] Faure C, Minder L. Cryptanalysis of the mceliece cryptosystem over hyperelliptic codes [C]//Eleventh International Workshop on Algebraic and Combinatorial Coding Theory, 2008:99 - 107.

[30] Bernstein D J, Lange T, Peters C. Attacking and Defending the McEliece Cryptosystem[C]//International Workshop on Post-quantum Cryptography. Springer-Verlag, 2008.

(上接第 301 页)

[6] 孙韩林, 刘建华. 公众网络统一身份认证服务及标准研究[J]. 电信科学, 2017, 29(2):84 - 88.

[7] 邓红, 杨茹, 王亚东. 基于云计算平台的鉴权分析[J]. 信息技术, 2014, 38(7):180 - 182.

[8] Owen S. Zxing[M]. Zebra Crossing, 2013.

[9] Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine learning in Python[J]. Journal of machine learning research, 2011, 12(10): 2825 - 2830.

[10] M'Raihi D, Machani S, Pei M, et al. Totp: Time-based one-time password algorithm[R]. RFC 6238, 2011.

[11] McCallum A, Nigam K. A comparison of event models for naive bayes text classification[C]. AAAI-98 workshop on learning for text categorization. 1998, 752: 41 - 48.

[12] 邓小鹏, 邢春晓, 蔡莲红. Web 应用测试技术进展[J]. 计算机研究与发展, 2007, 44(8):1273 - 1283.

[13] 宋巍, 张春柳, 邬斌亮. Web 系统性能测试研究与实践[J]. 计算机应用与软件, 2015, 32(3): 4 - 6.

[14] Memon P, Hafiz T, Bhatti S, et al. Comparative Study of Testing Tools Blazemeter and Apache Jmeter[J]. Sukkur IBA Journal of Computing and Mathematical Sciences, 2018, 2(1):70 - 76.