密码学报 ISSN 2095-7025 CN 10-1195/TN
Journal of Cryptologic Research, 2022, 9(1): 113–125

E-mail: jcr@cacrnet.org.cn
http://www.jcr.cacrnet.org.cn
Tel/Fax: +86-10-82789618

Android 应用内第三方支付协议的形式化分析*

李 晖, 范立岩, 潘雪松, 冯皓楠

北京邮电大学 网络空间安全学院, 北京 100876
通信作者: 李晖, E-mail: lihuill@bupt.edu.cn

摘 要: 应用内第三方支付具有便捷实用的特点, 使得众多的移动应用选择嵌入第三方支付功能, 但其安全性缺乏系统全面的分析, 导致支付安全难以保证. 为此, 对基于 Android 平台的五种第三方支付协议的协议流程、安全假设和安全目标进行形式化建模, 并采用 ProVerif 对协议进行分析, 分析结果表明, 协议在 Android 平台上的实现难以抵御订单篡改、订单替换和通知伪造等攻击; 其次, 在测试了 210 个采用应用内第三方支付的 App 后, 发现近 13.8% 的 App 均能被成功攻击; 最后, 为抵御上述攻击, 对第三方支付协议的实现提出了具体的建议.

关键词: 支付协议; 协议分析; 形式化分析; 应用内支付
中图分类号: TP309.1 **文献标识码:** A DOI: 10.13868/j.cnki.jcr.000507

中文引用格式: 李晖, 范立岩, 潘雪松, 冯皓楠. Android 应用内第三方支付协议的形式化分析[J]. 密码学报, 2022, 9(1): 113–125. [DOI: 10.13868/j.cnki.jcr.000507]
英文引用格式: LI H, FAN L Y, PAN X S, FENG H N. Formal analysis of third-party payment protocol in Android applications[J]. Journal of Cryptologic Research, 2022, 9(1): 113–125. [DOI: 10.13868/j.cnki.jcr.000507]

Formal Analysis of Third-party Payment Protocol in Android Applications

LI Hui, FAN Li-Yan, PAN Xue-Song, FENG Hao-Nan

School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China
Corresponding author: LI Hui, E-mail: lihuill@bupt.edu.cn

Abstract: Third-party in-app payment is convenient and practical, which makes many mobile applications choose to embed third-party payment function, while its security is not systematically and comprehensively analyzed, which means that, there is no guarantee on the payment security. To solve this issue, this paper gives a formal analysis on five third-party payment protocols based on Android platform by formalizing its security assumptions and security goals, and presents the modeling the protocol flows using ProVerif. Analysis results show that the implementation of the protocol on Android platform does not resist the attacks of order tampering, order replacement, and notification forgery. Moreover, after testing 210 apps that use third-party in-app payment protocols, 13.8% of them are found to be vulnerable to these attacks. In order to resist the above attacks, specific suggestions given on the implementation of third-party payment protocols.

Key words: payment agreement; protocol analysis; formal analysis; in-app payment

* 基金项目: 国家科技重大专项 (2018ZX03001010-005)
Foundation: National Science and Technology Major Program (2018ZX03001010-005)
收稿日期: 2021-02-24 定稿日期: 2021-05-28

1 引言

随着计算机、通信和互联网等多种技术的发展, 第三方支付手段逐渐被人们接受和普遍使用, 以支付宝 [1]、Paypal [2]、微信支付 [3] 和 Apple Pay [4] 为代表的应用内第三方支付蓬勃发展. 通过在移动终端应用中嵌入第三方支付平台的 SDK, 使得用户可以在应用内直接实现支付账单功能, 而不需要切换到其他应用或 web 浏览器, 目前这种支付方式已成为了主流的支付方式 [5].

应用内第三方支付协议是用户、商家和支付方为完成应用内支付功能进行的交互协议, 一般建立在密码体制基础上, 其安全性直接关系到支付能否安全顺利地完成 [6]. 分析表明, 集成应用内第三方支付的 App 面临伪造付款、隐私泄露等威胁 [7]. 特别是 Android 平台因其开放的特性, 以及大量的针对 Android 平台的 HOOK 框架和易用的攻击工具, 使基于 Android 平台的支付协议运行环境更具风险 [8].

目前针对应用内第三方支付协议的安全性分析工作大多依赖经验的直接观察法、根据已知的攻击方法对协议进行攻击测试和验证 [5,7]. 截止目前, 尚未发现采用形式化分析方法来对应用内第三方支付协议进行分析的成果. 而形式化分析方法基于数学定理和形式推理, 可以更全面的分析协议的安全性 [9,10], 如通过对 TLS1.3 协议 [11] 的分析, 发现攻击者能够在握手期间成功的模仿客户端, 通过对 5G-AKA 协议 [12–14] 分析, 发现协议在密钥保护方面未达到预定的安全目标, 通过对 EAP-AKA [15,16] 协议的分析, 发现存在中间人攻击等.

通过分析发现, 应用内第三方支付协议的形式化分析存在两个方面的困难:

(1) 应用内第三方支付协议缺乏明确的协议标准, 研究者难以对协议流程进行准确地建模. 与很多基于标准化的安全协议, 如 OAuth [17]、5G-AKA [12–14] 不同, 目前应用内第三方支付的的支付协议尚未标准化;

(2) 协议的安全目标模糊, 缺乏可量化的标准. 各个厂商提供的文档往往只规定了开发者需使用 API 接口和签名算法, 没有规定明确的安全目标.

本文提供了以下贡献:

(1) 本文选择了五个在中国使用广泛的第三方支付平台 [18]: 支付宝、微信支付、银联支付 [19]、京东支付 [20] 和招行一卡通 [21], 深入分析这些平台提供的开发文档和样例代码, 通过归纳和抽象的方法, 给出了应用内第三方支付的协议流程, 同时提出了协议的安全假设和安全目标.

(2) 本文对协议流程和安全目标进行形式化建模, 采用 ProVerif 工具对应用内第三方支付协议进行形式化分析, 分析表明在安全假设无法全部满足的情况下, 存在订单篡改、订单替换和通知伪造三种类型的攻击.

(3) 在对 210 个不同类别的 App 进行实际攻击验证后, 发现近 13.8% 的 App 存在被攻击的隐患.

接下来将在第 2 节描述应用内第三方支付协议的架构和流程; 在第 3 节分析协议的安全假设和安全目标; 在第 4 节对协议流程、协议的安全目标进行建模, 并对协议进行形式化分析; 在第 5 节根据形式化分析的结果, 寻找现实中的攻击实例, 并提出了提高应用内支付协议安全的建议, 最后也就是第 6 节, 将给出结论.

2 应用内第三方支付协议

在从架构和协议流程两个方面介绍应用内第三方支付协议之前, 为方便阅读, 表 1 列出术语表.

2.1 架构

如图 1 所示, 共有四个实体参与应用内第三方支付协议流程, 分别是商家的应用 (merchant app, MA)、第三方支付 SDK (third-party payment SDK, TP-SDK)、商家服务器 (merchant server, MS) 和第三方支付服务器 (cashier server, CS). 其中, MA 是安装在用户移动终端上的某商家应用客户端软件, 用户通过使用 MA 浏览、选择和购买商家的商品; TP-SDK 是集成在 MA 中的软件开发工具包, 实现了支持应用内第三方支付的各种签名及验证算法, 并完成在线付款流程; MS 维护存储与用户、商品及交易相关信

表 1 术语表
Table 1 Term list

缩略词	全称	释义
MS	merchant server	商家服务器
MA	merchant app	商家的应用
CS	cashier server	第三方支付服务器
TP-SDK	third-party payment SDK	第三方支付 SDK
odi	order information	订单申请, 包含商品名、商品数量等信息
tn	trade number	订单号, MS 为每笔订单分配的唯一标识
tri	transaction information	订单详细信息, 包含商品名、商品数量、总金额、下单时间等
S	sign	消息签名
SKMS	sign private key of MS	MS 的签名私钥
SKCS	sign private key of CS	CS 的签名私钥
pidn	prepay id number	支付流水号, 是 CS 为每笔订单分配的唯一标识
n	nonce number	随机数
ppw	pay password	支付密码
ri	result information	付款结果信息, 包含支付结果、支付金额、订单号、商品名、商品数量等

息的数据库, 并完成与 MA 和 CS 的交互; CS 记录支付信息和状态, 并通知 MS 支付完成. MA 不具备抵抗攻击者攻击的能力; TP-SDK 具备抵抗攻击者攻击的能力; MS 和 CS 具有较强的实体安全性. MA 和 MS、TP-SDK 和 CS 通过无线方式进行交互, 存在被攻击的隐患; MA 和 TP-SDK 通过进程间通信进行交互, 同样存在被攻击的隐患, MS 和 CS 通过服务器间的数据专线进行交互, 安全性较高.

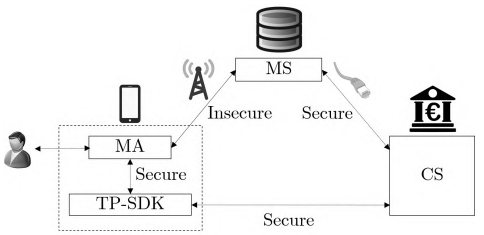

图 1 应用内第三方支付架构图
Figure 1 Architecture of in-app third-party payment

2.2 协议流程

在用户使用应用内第三方支付协议之前, 用户需要完成新用户注册及绑定第三方支付账号的相关流程, 从而使得用户具有在 MA 中购买并支付商品的能力. 随后, 用户在下订单后启动第三方支付协议流程.

应用内第三方支付协议可分为四个阶段: (1) 生成订单; (2) 绑定订单; (3) 在线付款; (4) 通知结果. 如图 2 所示.

(1) 生成订单: MA 向 MS 提交订单申请 (order information, odi), 包含商品名和商品数量信息.

(2) 绑定订单: MS 收到订单申请后, 进入绑定订单流程, 微信支付、银联支付和京东支付需要完成可选流程 (图 2 中用虚线框表示), 而支付宝和招行一卡通则无需执行可选流程. 在可选流程中, MS 采用自己的私钥对 tri 进行签名后传至 CS; CS 验签并校验后, 生成支付流水号 (prepay id number, pidn) 和随机数 (nonce number, n), 之后再对上述信息进行签名后传给 MS. MS 收到

可选流程中传过来的信息, 先验签, 再将 pidn 和 n 合并成 tri, 重新签名后传给 MA. 不执行可选流程的协议在 MS 收到订单后直接确认 MA 提交的新订单申请, 生成订单号 (trade number, tn) 以及订单详细信息 (transaction information, tri), MS 采用自己的私钥生成对 tri 的签名 S 后, 将 tri 和 S 传给 MA. MA 对收到的信息进行签名验证后, 向用户展示交易详细信息, 并将 tri 和签名 S 传至 TP-SDK.

(3) 在线付款: TP-SDK 收到信息后, 先进行 tri 信息校验, 成功后, TP-SDK 调应用内第三方支付收银台功能, 由用户输入支付密码 (pay pass-word, ppw), 完成付款.

(4) 通知结果: 付款成功后, CS 分别使用同步通知和异步通知两种方式通知付款结果. 其中同步通知流程为: CS 向 TP-SDK 返回由自己私钥签名后的付款详细结果 (result information, ri). TP-SDK 将消息传至 MA. MA 向用户展示 ri 后, 再将 ri 传至 MS, MS 进行签名验证、解析 ri 且核对正确后, 同步通知结束; 而异步通知流程为: CS 向 MS 返回由自己私钥签名后的 ri, MS 进行签名验证、解析 ri 且核对正确后, MS 完全校验 ri, 本次交易完成.

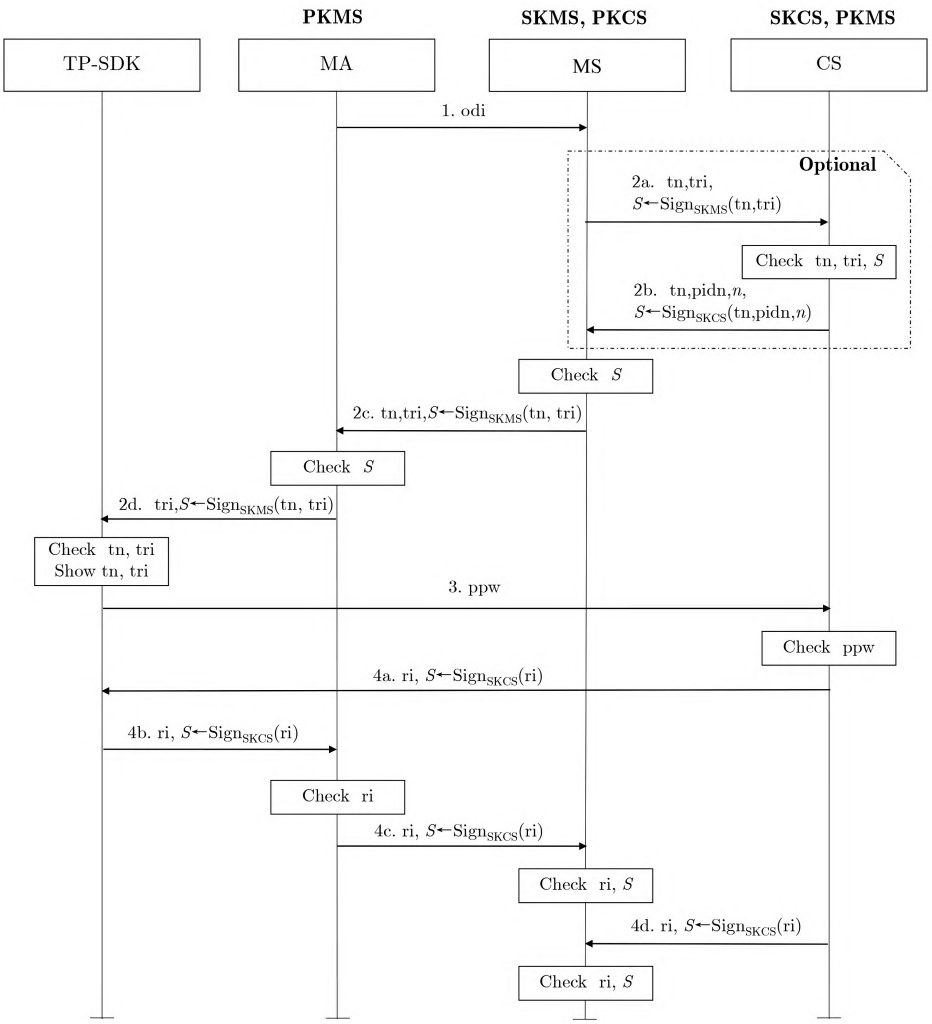

图 2 支付流程

Figure 2 Diagram of payment

3 安全假设与安全目标

在对第三方支付平台提供的开发文档进行充分分析后, 本节将描述确定的威胁模型和明确的安全目标, 这些都是采用形式化方法分析安全协议的前提和关键要素.

3.1 安全假设和威胁模型

参考微信支付给出的最佳安全实现范例 [22], 本文从密码算法、信道、数据保护和参与实体四个方面描述安全假设.

密码算法假设: 在应用内第三方支付协议采用的密码算法, 仅包括签名算法, 假设攻击者在不知道正确密钥的情况下, 无法伪造签名或者解密消息.

信道假设: 协议运行时的信道分成两类: (1) 公开信道: 攻击者可以在其中窃听, 拦截, 删除, 修改并且传输消息. (2) 私有信道: 攻击者无法将消息发送到此信道的任何参与实体, 也无法在该信道上窃听或者修改消息. 由于 MA 和 MS 之间的通讯没有采取任何保护措施, 因此, 假设其信道为公开信道; 而 MS 和 CS 通过数据专线交互, TP-SDK 和 MA 的通信采用了进程间通信的方式通讯, 由操作系统保证通信安全性, TP-SDK 和 CS 之间通讯的安全性由第三方支付平台采用 HTTPS 协议或私有协议、服务器防刷和 SO 加密等技术来保证, 因此, 假设 MA 和 MS、MS 和 CS、TP-SDK 和 CS 的通信信道为私有信道.

数据保护假设: 假设签名公钥是公开的, 而签名私钥是用户自己保存的秘密密钥, 不能被泄露.

参与实体假设: 下面通过分析实体可能被实际攻击的情况给出实体假设. 首先, TP-SDK 具备抵抗攻击者攻击的能力, 攻击者不能通过攻击实体获取信息或实施假冒攻击. 其次, MA 不具备抵抗攻击者攻击的能力, 实体被攻击后, 攻击者可以获得 MA 的源码和数据, 并可以对 MA 进行动态调试. 这种假设是合理的, 因为第三方支付厂商被要求采用安全加固和代码混淆等技术手段对 TP-SDK (内有关键安全算法) 进行保护, 但不强制对 MA 做与 TP-SDK 同等能力的安全防护. 最后, 本文主要研究来自移动终端的可能攻击, 因此, 假设 MS 和 CS 是具有实体安全性, 即攻击者不能获得 MS 和 CS 存储的数据.

综合上述各种假设, 考虑到协议尚未标准化、开发者开发 MA/MS 时存在对协议的简化、Android 平台中存在的密码学误用 [23] 等现实情况, 为了使分析结果可以覆盖更广的场景和便于分析, 本文将按照下面 5 种不同的安全假设条件分别对协议进行建模和分析:

安全假设 1: 假设参与实体 (MA、TP-SDK、MS 和 CS) 完整地实现了协议, MA 和 MS 之间使用了安全的通信协议, TP-SDK 具备抵抗攻击者攻击的能力, MA 不具备抵抗攻击者攻击的能力;

安全假设 2: 签名私钥被写入 MA, 且 MS 未完全校验 ri;

安全假设 3: 在安全假设 2 的基础上, tn 和 tri 由 MA 生成且未经 MS 签名;

安全假设 4: MA 和 MS 之间未使用安全通信协议 (如 HTTPS 协议), 且 MA 未完整展示支付信息;

安全假设 5: MS 省略了验签过程, 且 MS 未完全校验 ri.

3.2 安全目标

本文从机密性、认证性、完整性和不可否认性四个方面, 给出应用内第三方支付协议安全目标的形式化表述.

机密性: 任何情况下, 签名私钥 SKMS 和 SKCS 都应该保持机密, 否则, 消息签名机制将失去作用. 形式化地:

C1-签名私钥 SKMS 应始终保持机密

C2-签名私钥 SKCS 应始终保持机密

认证性: 应用内第三方支付协议应满足参与实体的认证性. 例如, MA 向 MS 发起新订单申请 odi, MS 接受 odi 并将生成的 tn 和 tri 签名并返回给 MA, MA 接收后验证 MS 签名, 这个过程可以看作是 MA 和 MS 的相互认证. 这里采用了 Lowe 的认证属性分类法 [24], 将实体 A 和 B 之间的认证属性由弱到强分为四种: 弱一致性 (weak agreement)—只确保 B 在之前已经运行过协议但不一定跟 A 一起运行; 存活性 (aliveness)—确保 B 曾与 A 一起运行协议但不一定使用相同的数据; 非单射一致性 (no-injective agreement)—确保 B 与 A 一起运行协议, 并对数据达成一致; 单射一致性 (injective agreement)—在非

单射一致性的基础上保证 B 的每次运行对应 A 的唯一运行. 该分类法广泛应用于协议的形式化研究 [12]. 形式化地:

A1-MS 必须在响应订单请求后与 MA 获得 odi 的单射一致性

A2-MA 必须在 MS 响应订单请求后与 MS 获得 odi 和 tn 的单射一致性

A3-MS 必须在确认支付结果后与 CS 获得 ri 的单射一致性

A4-MA 必须在确认支付结果后与 CS 获得 ri 的单射一致性

完整性: 任何情况下, 订单信息应保持完整, 否则, 攻击者可能会篡改订单申请和订单信息. 具体地: 在存在攻击者的情况下, odi 应在生成订单阶段保持完整, tn 应在绑定订单阶段保持完整, tri 应在绑定订单阶段保持完整. 形式化地:

I1-保证 MA 与 MS 在生成订单阶段获得 odi 的非单射一致性

I2-保证 MS 与 MA 在绑定订单阶段获得 tn 的非单射一致性

I3-保证 MS 与 MA 在绑定订单阶段获得 tri 的非单射一致性

不可否认性: 在任何情况下, 协议结束时能够分别给 MA 和 MS 提供有效的对方不可否认证据. 应用内第三方支付协议应保证 odi, tri, tn 和 ri 的不可否认性. 形式化地:

NR1-保证 MS 与 MA 在协议结束时获得 odi 的非单射一致性

NR2-保证 MA 与 MS 在协议结束时获得 tri 和 tn 的非单射一致性

NR3-保证 MA 与 MS 在协议结束时获得 ri 的非单射一致性

4 形式化分析建模

4.1 建模工具

本文使用 ProVerif 工具进行形式化分析. ProVerif 是一种自动的符号协议验证工具, 可以验证协议的机密性、认证性等安全属性 [25], 广泛应用于分析包括 TLS1.3 协议在内的诸多安全协议 [26–28]. 与其他形式化分析工具相比, 如 AVISPA [29] 和 Tamarin [30], ProVerif 解决了状态爆炸的问题, 且支持无限次数的会话. ProVerif 使用应用 pi 演算进行验证, 并基于 Horn 子句 [31] 的一阶逻辑解析规则. ProVerif 首先验证协议是否满足安全目标, 如果不满足安全目标, 则尝试自动构造攻击.

4.2 安全目标建模

对于**机密性**, attacker 语句可以直接质询攻击者是否可以获得机密. 例如, 质询攻击者是否可以获得 MS 的签名私钥:

```
query attacker(SKMS).
```

对于**认证性**, 首先定义与身份认证相关的事件类型, 然后质询事件之间是否存在对应关系. 例如, 事件 CreatOrderInfo(odi) 表示 MA 发出了 odi, 事件 MS_Generate_OutTradeNo(odi) 表示 MS 根据 odi 生成了订单号. 然后用以下方式质询这两个事件是否存在关于 odi 的单射一致性对应关系:

```
query odi : bitstring; inj-event(MS_Generate_OutTradeNo(odi))
    ==> inj-event(CreatOrderInfo(odi)).
```

对于**完整性**, ProVerif 没有提供直接验证完整性的语句, 但可以通过间接方式质询. 例如, 事件

MS_Generate_Transaction(odi, tn, tri) 表示 MS 根据 odi 和 tn 生成了 tri, MS_ALL_SUCCESS_Asyn(tn, tri) 表示 MS 相信订单号为 tn, 交易信息为 tri 的这笔交易已经成功了. 然后质询这两个事件是否存在非单射一致性的对应关系. 如果存在, 则证明在 tn 和 tri 未被篡改. 例如:

$$\text{query odi} : \text{bitstring}, \text{tn} : \text{tradeno}, \text{tri} : \text{transinfo}; \text{event}(\text{MS\_ALL\_SUC-CESS\_Asyn}(\text{tn}, \text{tri})) \\ ==> \text{event}(\text{MS\_Generate\_Transaction}(\text{odi}, \text{tn}, \text{tri})).$$

对于**不可否认性**, 在应用内支付的四个阶段中, 定义与不可否认性相关的事件类型. 例如, 在生成订单阶段, 事件 CreatOrderInfo_Fin(odi) 表示 MA 已经向 MS 发出了 odi, 事件 MS_Generate_OutTradeNo_Fin(odi) 表示 MS 收到了 MA 发出的 odi 且已经生成了订单号. 然后, 在 MA 发出订单后中注册事件 CreatOrderInfo_Fin(odi), 在 MS 接收订单申请并生成订单号过程后注册事件 MS_Generate_OutTradeNo_Fin(odi). 在最后质询这两个事件在生成订单阶段结束后是否均已发生:

$$\text{query odi: bitstring; event}(\text{MS\_Generate\_OutTradeNo\_Fin}(\text{odi})) \ \&\ \text{event}(\text{CreatOrderInfo\_Fin}(\text{odi})).$$

4.3 协议流程建模

参与协议的四个实体被建模成四个宏, 每个宏由描述协议流程的语句组成. 例如, let MA(odi: bitstring) 表示实体 MA 的宏, 参数 odi 表示 MA 的订单申请, out(HTTPS-Msg1, Msg1) 表示 MA 向 HTTPS-Msg1 信道发送 Msg1, in(HTTPS-Msg1, Msg2-A1: bitstring) 表示 MA 从 HTTPS-Msg2-A1 信道接收 Msg2-A1:

$$\begin{aligned} &\text{let MA(order\_info: bitstring)} = \\ &\text{let Msg1} : \text{bitstring} = \text{order\_info in} \\ &\text{out(HTTPS-Msg1, Msg1} : \text{bitstring);} \\ &\text{in(HTTPS Msg2-A1, Msg2-A1} : \text{bitstring);} \end{aligned}$$

为了更准确的对信道进行建模, 提升 Proverif 的分析效率, 实体间的信道建被模成多条逻辑子信道. 例如, MA 和 MS 之间的通信了三次, 则建模三条逻辑子信道, 信道 HTTPS-Msg1 传输消息 Msg1, 信道 HTTPS-Msg2-A1 传输消息 Msg2-A1, 以此类推:

$$\begin{aligned} &\text{free HTTPS-Msg1} : \text{channel.} \\ &\text{free HTTPS-Msg2-A1} : \text{channel.} \\ &\text{free HTTPS-Msg4-2} : \text{channel.} \end{aligned}$$

5 安全分析

在这部分将首先给出在五种安全假设条件下的应用内第三方支付协议的形式化分析结果, 之后给出了通过形式化分析找到的可能的攻击流程, 最后给出了对采用第三方支付协议的 App 进行攻击测试的结果.

5.1 形式化分析结果

表 2 为机密性和认证性的分析结果, 表 3 为完整性和不可否认性的分析结果, √ 表示满足安全目标, × 表示不满足安全目标, ! 表示部分满足安全目标. 可以看出, 除了在安全假设 1 的条件下, 应用内第三方支付协议都存在安全目标无法全部满足的情况.

ProVerif 构造了三种潜在类型的攻击: (1) 订单篡改攻击. (2) 订单替换攻击. (3) 通知伪造攻击. 在安全假设 2 或安全假设 3 下, 协议将遭受订单篡改攻击; 在安全假设 4 下, 协议将遭受订单替换攻击; 在安全假设 5 下, 协议将遭受通知伪造攻击.

表 **2** 机密性和认证性分析结果

Table 2 Analysis result of confidentiality and authentication

安全假设	C1	C2	A1	A2	A3	A4
安全假设 1	√	√	√	√	√	√
安全假设 2	×	√	√	!	!	!
安全假设 3	×	√	×	×	!	!
安全假设 4	√	√	×	×	√	×
安全假设 5	√	√	√	√	×	×

表 **3** 完整性和不可否认性分析结果

Table 3 Analysis result of integrity and non-repudiation

安全假设	I1	I2	I3	NR1	NR2	NR3
安全假设 1	√	√	√	√	√	√
安全假设 2	√	×	×	√	×	√
安全假设 3	√	×	×	√	×	√
安全假设 4	×	×	×	×	×	√
安全假设 5	√	√	√	√	√	×

5.2 可能的攻击

订单篡改攻击: 在安全假设 2 或安全假设 3 下, 安全目标 C1、I2、I3 和 NR2 未能达到. 对于 A1-A4, 协议没有达到单射一致性的认证性, 只能部分达到非单射一致性的认证性. 在上述安全目标未能达到时, 存在一种潜在的订单篡改类型的攻击. 在这种类型的攻击中, 攻击者充当恶意用户, MS 为受害者. 在这种情况下, 攻击者会篡改 tri, 并重新签名, 或者在本地生成订单的情况下, 直接篡改 tri, 并重新签名. 最终, 在商家不知情的情况下, 攻击者可以支付更少的钱. 图 3 和图 4 分别为两种订单篡改攻击的示意图.

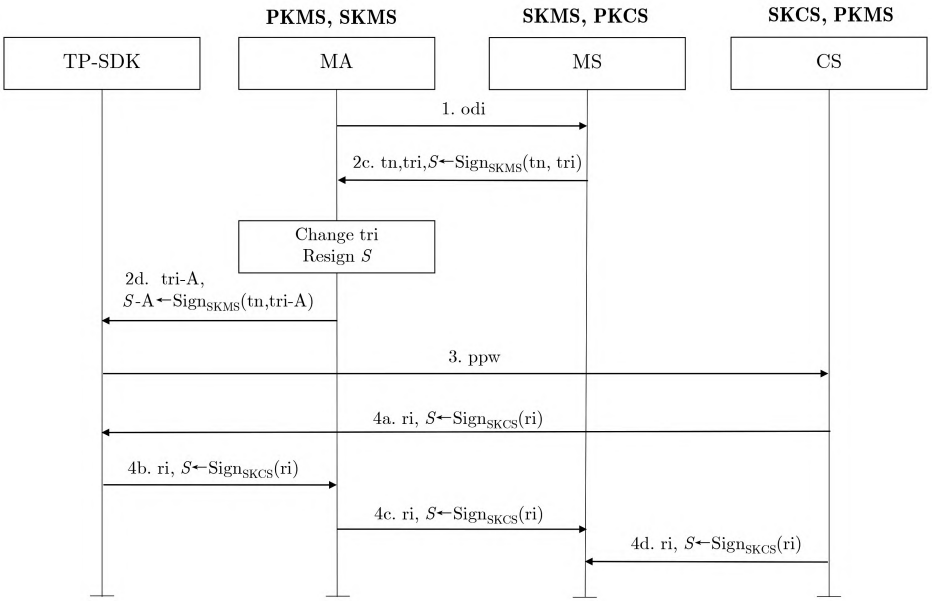

图 3 订单篡改攻击 1

Figure 3 Order tampering attack 1

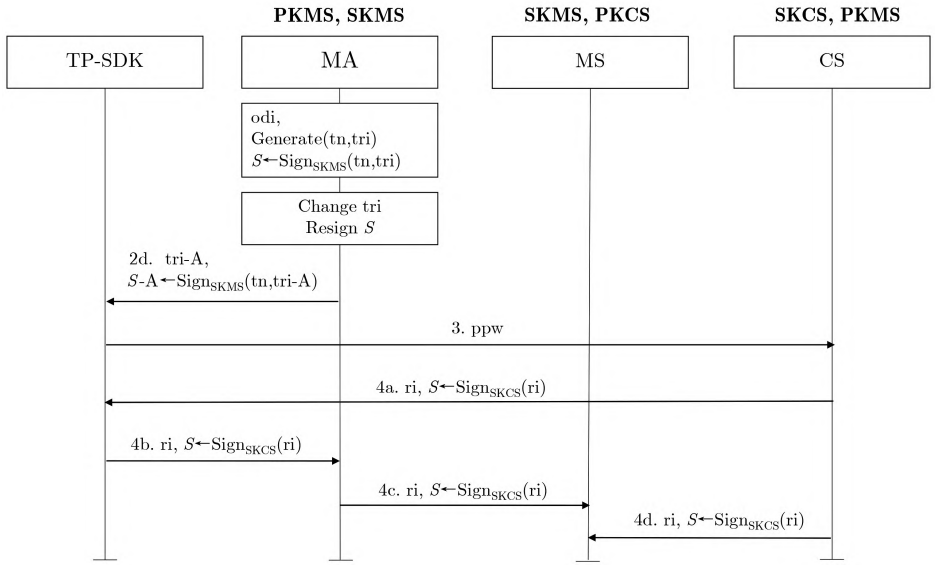

图 4 订单篡改攻击 2

Figure 4 Order tampering attack 2

订单替换攻击: 在安全假设 4 下, 安全目标 NR1、P2、A1、A2、A4、I1、I2 和 I3 未能达到. 在这种情况下, 存在一种中间人攻击. 在这种类型的攻击中, 攻击者充当中间人, MA 为受害者. 在这种攻击中, 攻击者将一笔交易的订单替换为另一笔交易, 并误导受害 MA 在不知不觉中为攻击者的订单付款. 当 MA 和 MS 的消息是通过不安全的网络通信通道传输时, 攻击者可以截取消息, 并用另一笔合法交易替代, 并将其发送给 MA. 然后, 受害者将为攻击者的订单付费. 图 5 为订单替换攻击的示意图.

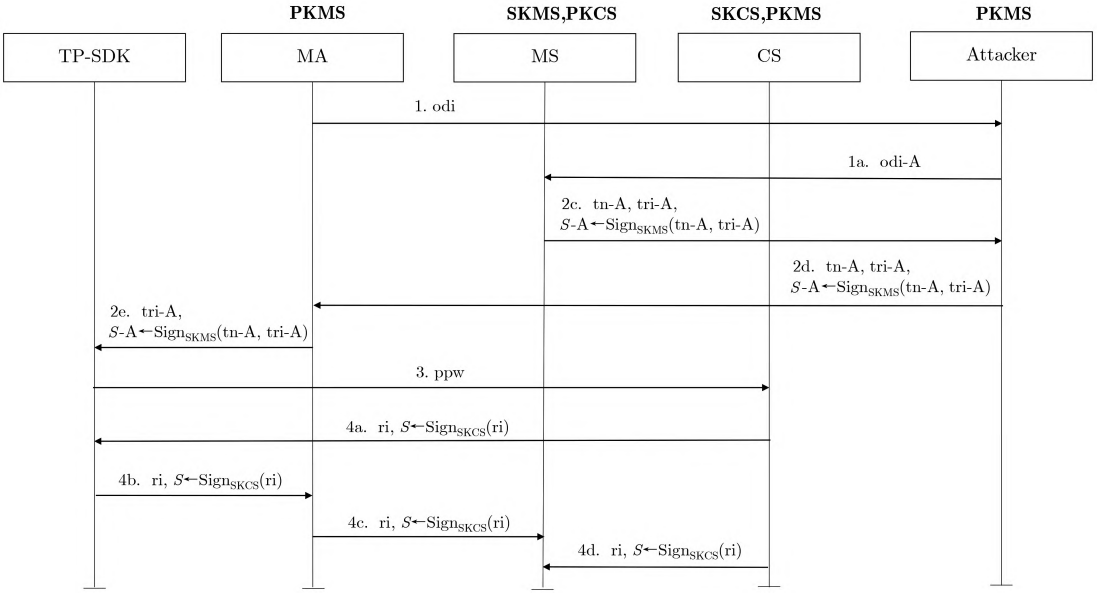

图 5 订单替换攻击

Figure 5 Order replacement attack

通知伪造攻击: 在安全假设 5 下, 安全目标 NR3、A3 和 A4 未能达到. 在这种情况下, 存在一种通知

伪造攻击. 在这种类型的攻击中, 攻击者充当恶意用户, MS 为受害者. 攻击者在 TP-SDK 要求用户确认订单并输入密码时不付款, 并向 MS 发送伪造的付款结果通知. 最终, 攻击者可以不付钱并得到商品. 图 6 为通知伪造攻击过程示意图.

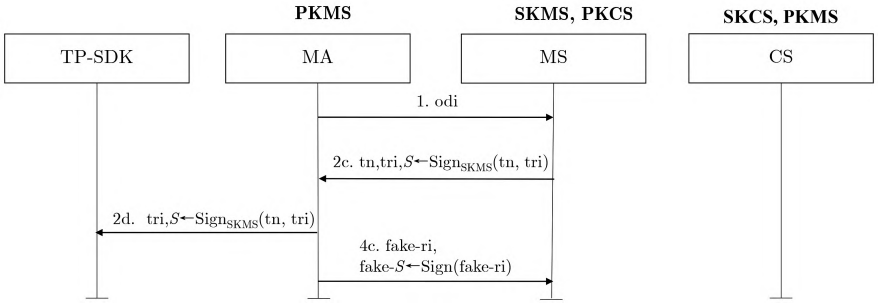

图 6 通知伪造攻击

Figure 6 Notification forgery attack

5.3 攻击验证结果

本文从腾讯应用宝批量下载了近 300 个 App, 涵盖了下载量榜单的头部中部和尾部. 经筛选, 共有 210 个 App 集成了应用内第三方支付. 经验证, 13.8% 的 App 存在形式化分析结果中发现的安全隐患, 这些 App 主要分布在下载量榜单的尾部. 共计 4 个 App 面临订单篡改的风险, 22 个 App 面临订单替换的风险, 3 个 App 存在通知伪造的风险. 表 4 为 App 检测结果汇总.

表 4 App 检测结果

Table 4 Result of app test

类型	测试数	潜在风险	订单篡改	订单替换	通知伪造
工具	41	3	1	2	0
购物	21	1	0	1	0
教育	57	10	1	8	1
文娱	72	12	2	9	1
游戏	19	3	0	2	1
共计	210	29	4	22	3

涉及网络通信的部分, 通过 burp suite 等工具搭建代理网络进行实验. 近 33.8% 的 App 在进行应用内第三方支付时使用 HTTP 通信, 这意味着在生成订单、绑定订单到和通知付款结果这三个阶段, 攻击者可以轻而易举的获得交易的详细信息. 4% 的 MS/MA 省略了验签过程; 1.43% 的 MS 消息未签名. 4.76% 的 App 存在 MS 未完全校验 ri 的情况.

对于 tn 和 tri 由 MA 生成、将签名私钥等放进 MA 这两种情况, 在使用 Apktool 和 IDA 对 App 逆向分析后, 发现部分 App 存在本地生成 tn 和 tri 的代码, 结合网络流量分析, 这些 App 在订单生成以及绑定订单到支付阶段, 或未与 MS 进行交互, 或将本地生成的 tn 和 tri 发给 MS, MS 签名后发回 MA. 约 13.8% 的 App 存在本地生成 tn 和 tri 的情况. 在静态分析的过程中, 发现 2.38% 的 App 存在将签名私钥写入 MA 的情况.

对于 MA 未完整展示支付信息, 则对 App 逐个进行人工分析, 检测支付时 App 是否将支付信息完整的展示给用户. 经检测, 36.2% 的 App 存在支付信息展示不全的情况, 这些 App 只展示了交易的总金额, 不展示订单号, 商品名和商品数量等信息. 表 5 为上述缺陷在样本 App 中的命中率.

需要指出的是, 不同于传统的购物 App, 很多工具类、教育类、文娱类和单机游戏类的 App 常常使用应用内第三方支付完成付费解锁某项功能, 可付款的内容一般被在 App 代码中进行限制, 而不是从服务器获取, 这些 App 更易于被攻击.

表 5 缺陷命中率
Table 5 Defect hit rate

缺陷	命中率 (%)
MA 和 MS 之间使用不安全的通讯	33.81
MA 未完整展示支付信息	36.19
MS/MA 省略了验签过程	2.86
MS 消息未签名	1.43
MS 未完全校验 ri	4.76
MA 生成 tn & tri	13.81
签名私钥放进了 MA	2.38

5.4 建议

本文的工作表明, 应用内第三方支付协议在 Android 平台进行实现的过程中, 存在多种无法保证应用内第三方支付的安全性的实现实例. 为了尽可能地规避这种情况, 本文给出了如下建议.

对于第三方支付平台:

(1) 应用内第三方支付协议应标准化. 目前, 大部分第三方支付平台只提供协议的简易流程图和 API 接口说明, 缺乏对协议的规范化描述, 且说明内容分散在各个文档中, 导致开发者难以准确地理解协议, 缺乏对协议的总体认识, 更多是在 “面向 API 接口” 开发.

(2) 明确应用内第三方支付协议的安全目标. 相比于 FIDO 协议、Oauth 协议和 5G-AKA 协议中明确的安全目标, 除微信支付对关于机密性的安全目标稍有提及外, 其他的第三方支付平台几乎没有对协议提出安全目标. 安全目标不是默认的、总所周知的, 而应由平台明确提出并规范化, 并涵盖机密性、完整性、认证性和不可否认性等安全属性.

对于应用:

(1) 需要实现 MA 与 MS 之间安全的网络通信, 例如使用 HTTPS 协议并正确配置 SSL 证书.

(2) 需能保证 MA 可以展示支付的详细信息, 包括本次支付的总金额, 支付的商品名和商品数量, 订单号等.

(3) 需要确保 MA 和 MS 的签名 & 验签过程不能被省略, 证书配置正确.

(4) 需要保证订单号 & 订单详细信息在 MA 提交订单申请后由服务器生成, 而不是由 MA 本地生成.

(5) 需要保证 MS 能够完成同步通知和异步通知, 并完成核对通知结果.

6 结束语

本文通过对应用内第三方支付协议进行形式化分析及对 Android 系统 App 进行实际攻击验证, 表明各种各样的原因使得应用内第三方支付协议常常运行在缺陷状态下, 导致协议无法达到其安全目标. 这些原因包括官方文档对安全目标的描述缺失、对协议流程的非标准化描述等使得开发人员更容易对协议的理解错误和简化开发, 最终使应用内第三方支付协议易于遭受攻击. 最后, 为保证应用内第三方支付协议的安全性, 本文对第三方支付平台和应用的开发者给出了建议. 在未来的工作中, 将重点从扩充安全假设、细化安全目标、扩大 App 攻击测试范围这三个维度继续深入研究.

参考文献

[1] Alpay open platform[EB/OL]. [2020-10-11]. https://open.alipay.com.
支付宝开放平台 [EB/OL]. [2020-10-11]. https://open.alipay.com.

[2] PayPal[EB/OL]. [2020-10-12]. https://www.paypal.com/c2/home.

[3] Weixin pay merchant platform[EB/OL]. [2020-10-11]. https://pay.weixin.qq.com.
微信支付开放平台 [EB/OL]. [2020-10-11]. https://pay.weixin.qq.com.

[4] Apple Pay[EB/OL]. [2020-10-12]. https://www.apple.com.cn/apple-pay/.
[5] YANG W B, LI J R, ZHANG Y Y, et al. Security analysis of third-party in-app payment in mobile applications[J]. Journal of Information Security and Applications, 2019, 48: 102358. [DOI: 10.1016/j.jisa.2019.102358]
[6] TELLEZ ISAAC J, SHERALI Z. Secure mobile payment systems[J]. IT Professional, 2014, 16(3): 36–43. [DOI: 10.1109/MITP.2014.40]
[7] YANG W B, ZHANG Y Y, LI J R, et al. Show me the money! Finding flawed implementations of third-party in-app payment in Android apps[C]. In: Proceedings of Network and Distributed System Security Symposium. 2017. [DOI: 10.14722/ndss.2017.23091]
[8] DONG Z J, WANG W, LI H, et al. SeSoa: Security enhancement system with online authentication for Android APK[J]. ZTE Communications, 2016, 14(S0): 44–50. [DOI: 10.3969/j.issn.1673-5188.2016.S0.005]
[9] XUE R,FENG D G. The approaches and technologies for formal verification of security protocols[J]. Chinese Journal of Computers, 2006, 29(1): 1–20. [DOI: 10.3321/j.issn:0254-4164.2006.01.001]
薛锐, 冯登国. 安全协议的形式化分析技术与方法 [J]. 计算机学报, 2006, 29(1): 1–20. [DOI: 10.3321/j.issn:0254-4164.2006.01.001]
[10] CHEN Y S, ZHAO J P, ZHU J M, et al. Formal protection architecture for cloud computing system[J]. ZTE Communications, 2014, 12(2): 63–66. [DOI: 10.3969/j.issn.1673-5188.2014.02.010]
[11] PATERSON K G, VAN DER MERWE T. Reactive and proactive standardisation of TLS[C]. In: Security Standardisation Research—SSR 2016. Springer Cham, 2016: 160–186. [DOI: 10.1007/978-3-319-49100-4_7]
[12] BASIN D, DREIER J, HIRSCHI L, et al. A formal analysis of 5G authentication[C]. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18). ACM, 2018: 1383–1396. [DOI: 10.1145/3243734.3243846]
[13] CREMERS C, DEHNEL-WILD M. Component-based formal analysis of 5G-AKA: Channel assumptions and session confusion[C]. In: Proceedings of Network and Distributed System Security Symposium. 2019. [DOI: 10.14722/ndss.2019.23394]
[14] KOUTSOS A. The 5G-AKA authentication protocol privacy[C]. In: Proceedings of 2019 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2019: 464–479. [DOI: 10.1109/EuroSP.2019.00041]
[15] ZHANG J J, WANG Q, YANG L, et al. Formal verification of 5G-EAP-TLS authentication protocol[C]. In: Proceedings of 2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC). IEEE, 2019: 503–509. [DOI: 10.1109/DSC.2019.00082]
[16] ZHANG J J, YANG L, CAO W P, et al. Formal analysis of 5G EAP-TLS authentication protocol using ProVerif[J]. IEEE Access, 2020, 8: 23674–23688. [DOI: 10.1109/ACCESS.2020.2969474]
[17] OAuth platform[EB/OL]. [2020-10-15]. http://oauth.net.
OAuth 说明文档 [EB/OL]. [2020-10-15]. http://oauth.net.
[18] Analysys-China Mobile Payment Report[EB/OL]. [2020-07-15]. https://www.analysys.cn/article/detail/20019826.
中国第三方支付移动支付市场报告 [EB/OL]. [2020-07-15]. https://www.analysys.cn/article/detail/20019826.
[19] Unipay open platform[EB/OL]. [2020-10-12]. https://merchant.unionpay.com/join/index.
银联支付开放平台 [EB/OL]. [2020-10-12]. https://merchant.unionpay.com/join/index.
[20] JDpay open platform[EB/OL]. [2020-10-12]. http://payapi.jd.com/docList.html?methodName=1#.
京东支付开放平台 [EB/OL]. [2020-10-12]. http://payapi.jd.com/docList.html?methodName=1#.
[21] CMBchina open platform[EB/OL]. [2020-10-12].http://openhome.cmbchina.com/PayNew/pay/doc/cell/app.
招行一卡通开放平台 [EB/OL]. [2020-10-12]. http://openhome.cmbchina.com/PayNew/pay/doc/cell/app.
[22] Best safety practices of Weixin-Pay[EB/OL]. [2020-10-15].
https://pay.weixin.qq.com/wiki/doc/api/app/app.php?chapter=23_3&index=4.
微信支付最佳安全实现范例 [EB/OL]. [2020-10-15].
https://pay.weixin.qq.com/wiki/doc/api/app/app.php?chapter=23_3&index=4.
[23] ZHANG T L, ZHANG Y Y, WANG H, et al. An empirical analysis of cryptographic misuse on different platforms[J]. Computer Science and Technology, 2017: 316–324. [DOI: 10.1142/9789813146426_0037]
[24] LOWE G. A hierarchy of authentication specifications[C]. In: Proceedings of 10th Computer Security Foundations Workshop. IEEE, 1997: 31–43. [DOI: 10.1109/CSFW.1997.596782]
[25] BLANCHET B. Modeling and verifying security protocols with the applied Pi calculus and ProVerif[J]. Foundations and Trends in Privacy and Security, 2016, 1(1–2): 1–135. [DOI: 10.1561/3300000004]
[26] BLANCHET B, CHAUDHURI A. Automated formal analysis of a protocol for secure file sharing on untrusted storage[C]. In: Proceedings of IEEE Symposium on Security and Privacy (SP). IEEE, 2008: 417–431. [DOI: 10.1109/SP.2008.12]
[27] BHARGAVAN K, BLANCHET B, KOBEISSI N. Verified models and reference implementations for the TLS 1.3

standard candidate[C]. In: Proceedings of 2017 IEEE Symposium on Security and Privacy (SP). IEEE, 2017: 483–502. [DOI: 10.1109/SP.2017.26]

[28] ABADI M, BLANCHET B, FOURNET C. Just fast keying in the pi calculus[J]. ACM Transactions on Information and System Security, 2007, 10(3): 9. [DOI: 10.1145/1266977.1266978]

[29] ARMANDO A, BASIN D, BOICHUT Y, et al. The AVISPA Tool for the automated validation of internet security protocols and applications[C]. In: Computer Aided Verification—CAV 2005. Springer Berlin Heidelberg, 2005: 281–285. [DOI: 10.1007/11513988_27]

[30] MEIER S, SCHMIDT B, CREMERS C, et al. The TAMARIN prover for the symbolic analysis of security protocols[C]. In: Computer Aided Verification—CAV 2013. Springer Berlin Heidelberg, 2013: 696–701. [DOI: 10.1007/978-3-642-39799-8_48]

[31] WEIDENBACH C. Towards an automatic analysis of security protocols in first-order logic[C]. In: Automated Deduction—CADE-16. Springer Berlin Heidelberg, 2016: 314–328. [DOI: 10.1007/3-540-48660-7_29]

作者信息

李晖 (1972–), 吉林长春人, 副教授. 主要研究方向为移动安全、协议安全.
lihuill@bupt.edu.cn

范立岩 (1996–), 黑龙江哈尔滨人, 硕士. 主要研究方向为移动安全、协议安全.
fanliyan-china@bupt.edu.cn

潘雪松 (1995–), 河南固始人, 硕士. 主要研究方向为移动安全.
panxuesong@bupt.edu.cn

冯皓楠 (1996–), 山西大同人, 硕士. 主要研究方向为移动安全.
fenghaonan@bupt.edu.cn