计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(10):61−67 [doi: 10.15888/j.cnki.csa.007131]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于微信的数据安全自毁服务平台①

李　凌[1], 刘子豪[2]

[1](河海大学 计算机与信息学院, 南京 211100)
[2](江苏科技大学 计算机学院, 镇江 212003)
通讯作者: 刘子豪, E-mail: 384015085@qq.com

摘　要: 针对社交软件中数据信息易泄露的现状, 展开了社交软件数据安全自毁服务平台的研究. 利用微信, 结合TLS 协议, 一种基于网络的数据自毁方法以及微信的第三方网页授权 OAuth2.0 机制构建了基于微信的数据安全自毁服务平台, 以微信作为基础推广平台, 用 TLS 协议保障客户端和服务器之间的数据传输的安全, 核心为改进后的一种基于网络的数据自毁方法作为服务器端的数据安全自毁方案, 最后使用微信的第三方网页授权 OAuth2.0 机制为基础完成对用户身份的识别. 通过结合实例与理论分析, 验证了设计方案的可行性和安全性.

关键词: 微信; 数据安全自毁; 服务平台; 身份识别

引用格式: 李凌,刘子豪.基于微信的数据安全自毁服务平台.计算机系统应用,2019,28(10):61–67. http://www.c-s-a.org.cn/1003-3254/7131.html

Secured Data Self-Destructive Service Platform Based on WeChat

LI Ling[1], LIU Zi-Hao[2]

[1](College of Computer and Information, Hohai University, Nanjing 211100, China)
[2](School of Computer Science, Jiangsu University of Science and Technology, Zhenjiang 212003, China)

Abstract: Facing precarious status quo of that users' information data in social software is easily leaked, we have carried out a research on secured self-destructive service platform of social software for data. We make full advantage of WeChat, combining TLS protocol, a self-destructive method based on network and OAuth2.0, an open protocol to allow secure authorization from the third party webpage of WeChat to construct a secured self-destructive service platform based on Wechat. We ensure the safety of data transmission between client and server by TLS protocol, take a modified method based on network as the secured self-destructive scheme for data on the server side and make the advantage of OAuth2.0 which is a secure authorization from the third party webpage of WeChat to accomplish users' identification. theories and example analysis verifies the feasibility and safety of this design plan

Key words: WeChat; secured data self-destruction; service platform; identification

随着移动互联网技术的兴起, QQ、微信、微博等一系列移动互联网社交平台成为年轻人的新宠, 越来越受人们的喜爱. 在日常的学习、生活乃至工作中, 我们都可以方便利用这些平台完成和他人的信息交流, 大大节约了交流成本. 社交软件中消息的通信主要由社交软件的服务器先进行保存然后转发给目标用户, 这不可避免在交流的过程中会在软件的服务器上留下长期的数据副本, 一旦服务器产生漏洞将有可能导致用户大量历史数据副本的泄露. 本文根据目前对移动端数据安全自毁的需要, 充分发挥微信现有用户量大, 推广方便的特点, 以一种基于网络的数据自毁方法[1]作为服务器端的数据安全自毁基础方案进行改进, 使用嵌入在微信公众平台中的网页发送原文件, 利用客户端以及服务期端的通信安全由 TLS 协议保证并结合基

① 收稿时间: 2019-02-26; 修改时间: 2019-04-17; 采用时间: 2019-05-06; csa 在线出版时间: 2019-10-15

于微信的第三方网页授权 OAuth2.0[2]机制验证用户的身份构建出可以为移动端用户提供基于微信的数据安全自毁的服务平台.

1 微信公众号及相关技术简介

微信是腾讯公司于 2011 年 1 月 21 日推出的一款手机应用. 微信用户可以通过软件与好友分享文字与图片, 并且支持语音、视频等多功能的服务. 微信公众平台是腾讯公司基于微信基础平台上新增的模块, 每个 QQ 账号都可以打造属于自己的微信公众号, 微信公众平台可以为特定群体提供自定义的服务. 随着微信与手机号的绑定, 各种各样基于微信身份的互联网应用服务随之产生. 为了有效的获取到微信用户的基本信息, 微信推出了第三方网页授权 OAuth2.0 机制.

微信网页授权 OAuth2.0 机制是用户在微信中访问第三方网页时, 公众号通过网页授权取得用户基本信息, 进而实现业务逻辑的机制. 第三方网页通过该机制获取到用户信息的前提有 2 个. 第一, 授权的第三方页面需提前在微信开发者中心进行域名配置. 第二, 被获取信息的用户必须在 48 小时内和公众号有过最基本的数据交互, 并且用户处于关注该公众号的状态. 当用户点击处理过的第三方网页链接时, 会首先自动跳转到微信授权中心确认该链接是否是在指定的微信默认游览器中打开以及链接的域名是否提前在微信的开发者中心配置过. 若通过微信授权中心的验证, 则返回当前用户 OpenID 所对应的一次性 code, 作为参数传入重定向的业务页面. 在业务页面中, 服务商使用一次性 code 可以换取到对应的用户包括 OpenID 在内基本信息, 确保用户身份的真实有效.

2 相关工作

微信具有用户量大, 服务平台使用方便, 推广迅速的特点. 目前, 社交软件中带有敏感数据的数据的泄露主要有以下两个主要途径: 1、带有敏感信息的数据在传输中泄露. 2、存储在服务器数据库上的数据副本被窃取. 由于社交软件普遍使用 HTTP 协议进行通信, 虽然对通信的消息内容部分进行过一定的加密处理, 但其通信安全完全依赖于加密密钥的安全, 一旦密钥泄露将导致通信安全的下降. 并且在传递的过程中, 数据都是以副本的形式进行传递, 其不可避免的会在服务器上留下数据的副本. 在服务器内, 为方便调用用户的数据, 一般都会设置访问权限, 然而从乌云网上可以发现由于服务器漏洞的频发以及部分业务逻辑设计的不合理, 攻击者常常可以非法的获取访问权限或者绕过访问权限, 获取到大量的用户历史数据副本.

微信自发布运行至今, 安全性问题时有发生. 2014 年发生了所谓的视频泄露事件, 后腾讯官方证实并非微信本身存在漏洞, 但专家表示在微信内容分享机制中确有考虑不周之处. 2016 年 8 月, 采用了 X5 内核的微信被证实存在高危安全漏洞, 该技术漏洞将允许黑客取得微信的完全控制权, 获取包括微信隐私、聊天记录、微信钱包等在内的信息[3]. 谁也无法保证今后不会再次出现类似的安全性问题. 聊天记录, 账单信息这些数据是否需要持久化存储? 某些数据用户因为隐私不希望持久记录且不被任何人得知, 希望实现真正的“阅后即焚”. 但是这些记录在“删除”后仍然可以通过技术手段还原, 因此数据的自毁就显得很有必要.

本文为解决上述存在的问题, 以一种基于网络的数据自毁方法作为服务器端的数据自毁基础方案进行改进, 利用 TLS 协议保证通信传输中的数据安全, 结合微信的第三方网页授权 OAuth2.0 机制完成数据安全自毁服务平台的构建. 自毁服务器以公众号为媒介, 位于用户端和微信服务端之间. 用户通过微信服务器提供的身份验证功能登录公众号, 利用公众号为媒介传递数据, 自毁服务器为公众号提供加密解密以及自毁等功能.

3 基于微信的数据安全自毁服务平台设计

3.1 系统信息流转过程

基于微信的数据安全自毁服务平台的数据流转过程如图 1 所示. 发送方用户通过微信公众平台的自定义菜单打开数据上传页面, 按照规定填写好所需要的数据. 将所有的相关数据直接发送给数据安全自毁服务器, 传输过程的安全由 TLS 协议保障. 数据安全自毁服务器利用接受到的数据对原数据进行加密分片处理, 返回的 SDD 代理链接[4]包装成微信规定的图文消息的 XML 格式发送给微信服务器, 如图 2. 由微信服务器通过客服接口直接转发给接收方用户. 接收方用户通过点击接收到的图文消息的方式完成身份的验证后, 直接从数据安全自毁服务器上获取到原数据. 图文消息封装格式如表 1.

数据流传输过程如下, 用户首先登录微信服务器, 进行身份验证, 获取接收方用户唯一的身份标识, 并将其加入进数据流中. 之后发送方将需要发送的数据通

过微信公众号的方式发送到自毁服务器, 服务器通过具体的加密算法对数据进行加密, 并将其存储到数据安全自毁数据库中. 对于接收方, 首先通过微信服务器进行登录, 来验证身份, 这样就可以与自毁服务器中存储的身份标识进行匹配, 进而从自毁数据库中读取数据片段. 之后自毁服务器通过相关的秘钥来读取和整合拼接相关的数据. 对于已阅, 过期或者删除的数据, 将不会存储在数据安全数据库中.

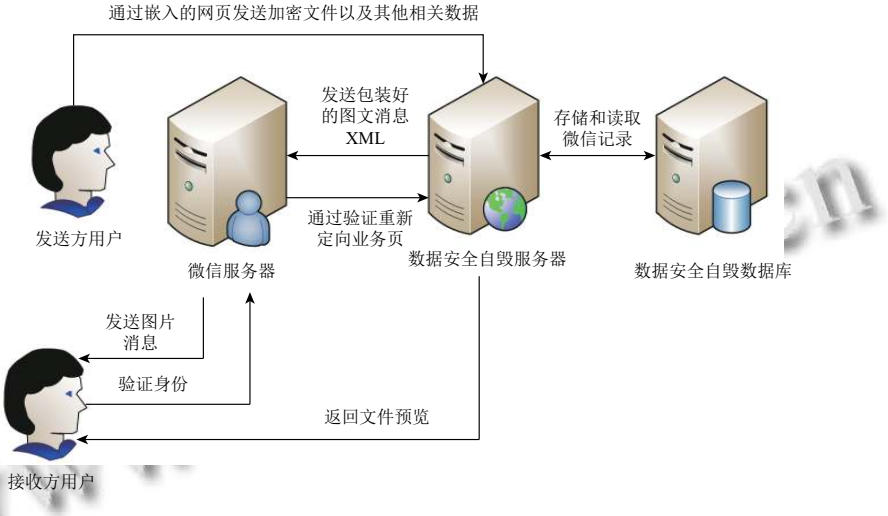

图 1 系统信息流程表

```
<xml>
    <ToUserName><![CDATA[toUser]]></ToUserName>
    <FromUserName><![CDATA[fromUser]]></FromUserName>
    <CreateTime>12345678</CreateTime>
    <MsgType><![CDATA[news]]></MsgType>
    <ArticleCount>2</ArticleCount>
     <Articles>
        <item>
            <Title><![CDATA[title1]]></Title>
            <Description><![CDATA[description1]]></Description>
            <PicUrl><![CDATA[picur1]]></PicUrl>
            <Url><![CDATA[url]]></Url>
        </item>
         <item>
            <Title><![CDATA[title]]></Title>
            <Description><![CDATA[description]]></Description>
            <PicUrl><![CDATA[picurl]]></PicUrl>
            <Url><![CDATA[url]]></Url>
        </item>
    </Articles>
</xml>
```

图 2 XML 文件

表 1 参数描述

参数	视频
ToUserName	接收方帐号
FromUserName	开发者微信号
CreateTime	消息创建时间 (整型)
MsgType	news
ArticleCount	图文消息个数, 限制为 10 条以内
Articles	多条图文消息信息
Title	图文消息标题
Description	图文消息描述
PicUrl	图片链接
Url	点击图文消息跳转链接

3.2 数据加密模型

本文利用珠链模型 (Bead Strand Model), 将数据分为两个部分, 珠 (Beads) 和链 (Strand). 如图 3 所示, 珠被一根特定的链所连接. 该模型与单纯的链表不同, 它是一个面向流数据的存储结构, 不仅仅提供了数据持久化, 还涵盖可众多的数据类型.

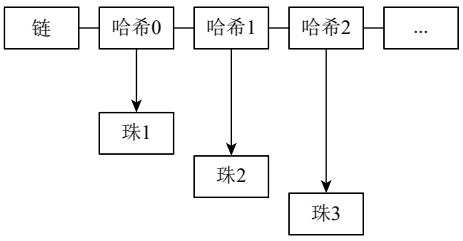

图 3 珠链模型结构

利用该模型对数据流进行加密, 并存储进自毁服务器数据库中. 其过程分为 4 个步骤:

第一步, 产生一个随机值作为 *key*, 记作 *RandomKey*, 之后将输入的数据和 *RandomKey* 结合一起进行加密, 记作 *CipherData*. 通过使用对称加密算法, 获得加密数据. 本文具体使用了 AES 加密算法[5], 该算法使安全性和有效性得到了平衡.

第二步, 计算 *CipherData* 的哈希值, 得到 *CipherHash*. 之后将 *CipherData* 拆分成不大于 8 KB 的片段, 分别为 *Piece 1*, *Piece 2*, *Piece* 3 等多个片段. 之后将 *RandomKey*, *CipherHash* 和 *CipherData* 的长度合并成一个片段, 记作 *Piece 0*. 如果任何一个片段的长度不足 8 KB, 不足的部分用随机字节补充. 选择 8 KB 的理由在于, 在 java 运行环境中, 内存池能够分配的最小缓冲单位默认为 8 KB.

第三步, 为每一个 8 KB 片段产生随机 UUID (version 4 UUID)[6], 并为每一个片段加上时间戳标签, 记作 *ExpirationTime*. 每一个珠, 都由 UUID、其对应的片段和 *ExpirationTime* 构成. 使用分布式数据库来进行持久化存储. 通过对比 *ExpirationTime* 和当前的网络时间, 此数据库会删除过期或者失效的珠. 对于 *ExpirationTime*, 用户可以自行设置. 但是如果同一条数据的所有珠存在同一个具体 *ExpirationTime*, 这会导致一个后果: 攻击者如果获取了数据库资料, 就可以很轻松的区分哪些珠子属于同一条数据. 因此一种延迟机制被采用来降低这种风险: 给每一个 *ExpirationTime* 随机延时延迟 0 到 3600 秒, 这样可以提高复杂度, 并使得珠之间有足够的离散.

第四步, 将片段中的 UUID 按顺序结合起来, 获得由 uuid 组成的线, 称作链, 任何拥有链的用户可以获得所有的片段, 通过 *RandomKey* 破译 *CipherData*, 通过 *Piece* 0 中的 *CipherHash* 破译数据. 但是如果任何珠过期或者被删除, 即使你获得了相应的链, 你也无法获得正确的珠以及数据.

以上 4 步的伪代码展示如下:

算法. 数据加密与解密

```
RandomKey = Generate ();
CipherData = Encrypt (RandomKey, InputData);
CipherHash = Hash (CipherData);
Piece[0] = Combine (RandomKey, CipherHash, Lengthof
(CipherData ) );
For n=1 To <Lengthof (CipherData )/(1024*8)>
Piece[n]= Split (CipherData, (n-1)*1024*8, (n)*1024*8);
End For
For Each Piece in Piece[n]
UUID = GenerateUUID ();
CreateBead(UUID,Piece,ExpirationTime.addSeconds(
RandomDelay.ToSeconds ( ) ) );
Strand = Strand + UUID;
End For
Strand = “SDD://” + Strand;
Return Strand;
```

其中 SSD (self-destructing data) 链接将暴露出作为 URL (Uniform Resource Locator), 因此用户可以以一个<img>标签的形式将其嵌入任何基于 HTML (HyperText Markup Language) 的文档中. 任何收到 SSD 链接的人可以使用支持 HTML 的 Web 浏览器来阅读 SSD 信息. 原型系统架构展示如图 4.

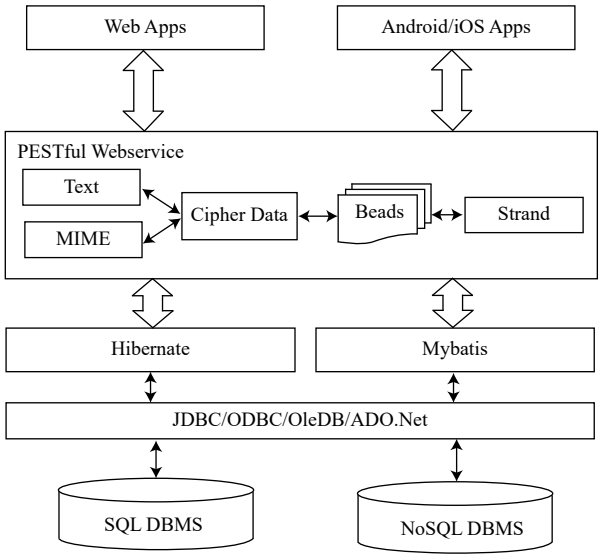

图 4 原型系统架构

但是 HTML 文本依赖于 HTTP (HyperText Transfer Protocol)[7]中的 GET 方法, 且最大长度为 2048 个 ASCII 字符[8]. 这意味着我们无法显示一个 2054 个字符的链作为 HTML 文本.

Geambasu 使用火狐插件将密码数据转成纯文包[9], 基于此也可以将 SSD 形式的 URI 转换成解密文本[10], 但是在公众号场景下这种方式不适用. 因此, 我们设计了一种二级索引机制, 当链创建成功后, 对应生成一个单个的 UUID 用来做映射. 用户可以通过此 UUID 和链来取得数据.

总的来说, 发送方通过这种模型将数据加密存储, 并获得相应的链. 接收方通此链, 可以对离散的数据片段进行重新整合, 从而达到解密的目的. 解密之后, 数据会被判定为过期数据, 从而在存储端被删除. 微信服务端提供的是身份验证, 具体的加密解密工作由自毁服务器来完成. 发送方和接收方通过公众号为平台进行安全通信. 可以将自毁服务器想象成一个盒子, 一个人将要传递的数据撕成碎片放进盒子里, 盒子会对应生成一把钥匙和一段咒语. 当另一个人拿着钥匙将其打开后, 看到只是一地碎纸, 只有当他念动咒语后, 碎纸才会自动整合成一段完整的数据, 并在之后燃烧成灰.

3.3 数据安全自毁业务流程设计

数据安全自毁存储流程如图 5 所示, 系统从发送方用户获取到四项数据, 过期时间 TOE、接收方用户唯一标示符 OpenID、原数据以及发送方的用户唯一标识符所对应的一次性 code 值. 其中前三项数据由发送方用户填写, code 值则通过微信特有的第三方网页授权 OAuth2.0 机制获取, 保证发送方用户身份的真实有效. 首先, 使用接收方用户唯一标识符 OpenID 作为密钥对文件数据进行 AES 加密. 完成一次加密后, 使用随机密钥函数产生的密钥对密文进行 2 次 AES 加密, 对加密后的密文进行分片并将密钥伪装成一个密文分片, 分片大小为 1 K. 随机选择一个分片添加时间戳属性 TOE, 其他分片添加时间戳属性.

TOE+random (3600*24*1000). 由于文件数据还原需要所有的密文分片, 此时间戳分配方案可以确保在用户设定的过期时间后文件无法再被还原. 同时同一文件密文分片之间的时间耦合性下降, 防止可能的利用同一文件不用分片时间戳相同这一特性进行分析和攻击. 给每个密文分片记录产生唯一 UUID, 并将其作为记录的主键存入服务器的数据库中. 根据分片的顺序将对应的 UUID 进行 2 次哈希后组装成 SDD 代理链接, SDD 包装成微信规定的图文消息的 XML 格式发送给微信服务器, 由微信服务器通过客服接口直接转发给接收方用户. 用一开始接收到发送方的用户唯一标示对应的 code 值向微信服务器换取发送方的 OpenID 标识数据来源. 服务器定时扫描数据库, 删除时间戳早于当前时间的分片记录.

数据安全自毁读取流程如图 6 所示, 接收方用户通过点击图文消息向服务器发送解密请求, 服务器从接收方用户获取到指定 SDD 代理链接以及接收方身份所换取的对应的一次性 code 值. code 值通过微信特有的第三方网页授权 OAuth2.0 机制获取, 保证接收方用户身份的真实有效. 使用一次性 code 值向微信授权中心换取对应用户的 OpenID. 服务器根据接收到的 SDD 代理链接向服务器数据库申请相对应的密文分片, 若分片都没有过期被删除则按照顺序读取所有的分片使用随机密钥进行的 AES 解密. 使用之前换取到接收方用户的 OpenID 作为密钥进行 2 次 AES 解密, 解密成功则根据文件的类型向用户提供在线预览服务.

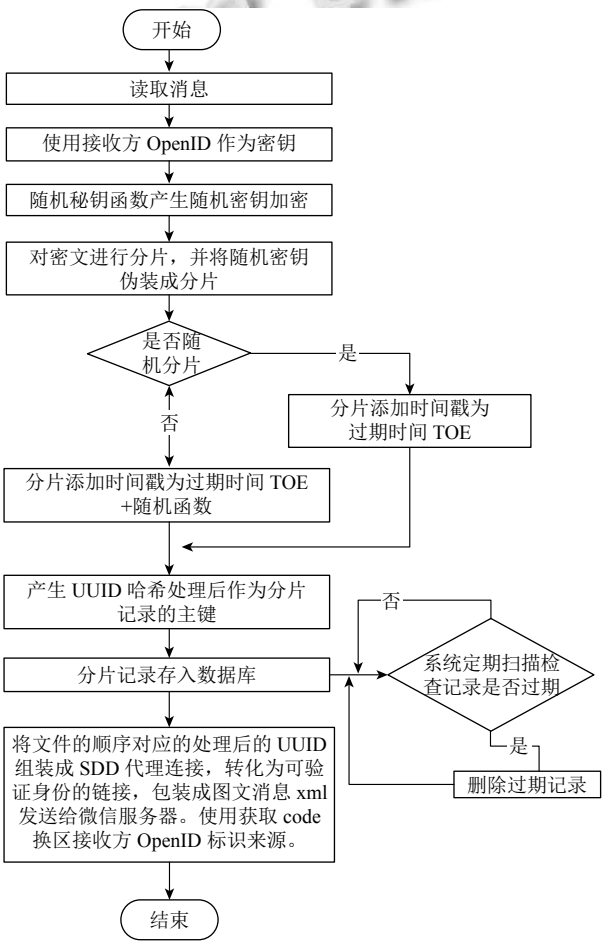

图 5 数据安全自毁存储流程

3.4 应用实例

通过微信公众号发送安全自毁文件如图 7 所示, 用户由微信公众号下单的自定义菜单的上传文件的选项进去上传文件页面. 一共有 3 项数据要求用户填写, 分别是用户所设定的文件安全自毁时间, 所需要安全自毁的文件以及接收方用户的 OpenID. 正确填写后, 点击发送, 若返回文件发送成功的提示则发送成功.

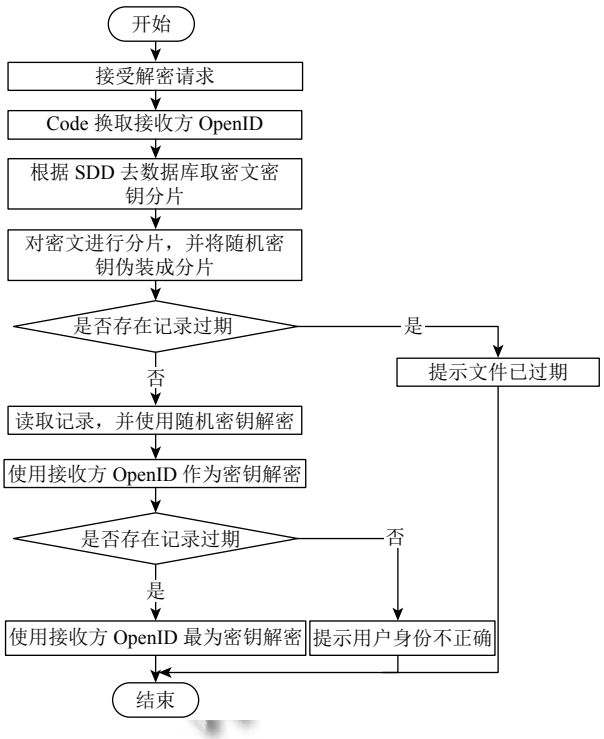

图 6 数据安全自毁读取流程

图 7 微信公众号文件发送页面

通过微信公众号接受安全自毁文件如图 8 所示, 接收方用户接收到文件, 点击图文消息, 微信自动验证当前用户身份, 若用户身份正确且文件没有超过过期时间, 则可以获取到文件的在线预览.

4 系统的安全性分析

4.1 在微信公众号中的安全性分析

在数据安全自毁服务平台设计方案中, 文件数据本身只与数据安全自毁服务器进行直接交互, 并不会经由微信服务器进行转发, 也就不会在微信中留下长期的数据副本. 客户端以及数据安全自毁服务器之间的通信安全由 TLS 协议所保证, 安全性高于微信所使用 HTTP 的通信. 在微信的通信中, 由文件的 SDD 代理链接作为解密凭证代为传递. 在用户试图使用 SDD 代理链接作为解密凭证获取原文件时, 我们使用基于的微信第三方网页授权 OAuth2.0 机制的来获取用户的身份, 并将其 OpenID 做为解密密钥使用. 当文件过期时间到达时, 数据安全自毁服务器会删除保存在数据库中过期的密文分片, 此时 SDD 代理链接失效, 原文件再也无法再被还原, 有效的防止了可能的大量历史文件的泄露.

图 8 微信公众号文件发送页面

接收方用户的图文消息都是直接由微信公众平台以客服消息的模式所发送, 唯一的身份标识符为双方用户在该微信公众号中所特有的 OpenID. 由于微信公众号 OpenID 的特性, 即不同用户对同一微信公众号的 OpenID 不同, 同一用户对不同的微信公众号 OpenID 也不同, 攻击者无法通过 OpenID 获取到接收方或者发送方的其他个人信息, 而获取 SDD 链接需通过微信的身份验证机制才能正确的读取文件, 所以即使微信发送消息被攻击者截获分析也并不会对用户造成任何安全性问题.

4.2 在数据安全自护服务器端上的安全性分析

在用户设定的过期时间内, 文件数据受高安全性的加密方案所保护. 而当有效期到达时, 文件的密文分片会被服务器端数据库自动删除, 文件无法再通过

SDD 代理链接被还原. 下面给出过期时间内, 随机预言模型[11]下加密方案安全的简单证明.

假设加密明文所用的随机密钥产生函数是一个随机预言机, 则攻击者无法利用该预言机的弱点预测每次加密所用的密钥, 即该预言机所产生的随机密钥在其候选值域内均匀分布. 在无法预测密钥的情况下, 由于该加密方案中明文加密后的密文被分片成等长的大小并且密钥被伪装成一个密文分片, 所以服务器攻击者将无法区别哪些分片是密文哪些分片是密钥. 假设密钥数量远小于密文碎片数量, 则认为单个分片在概率上最有可能是密文碎片, 此时攻击者要寻找到密钥相当于只能暴力破解. 所以对单个文件的密文密钥分片来说 O(n) 是穷举加密算法密钥空间的时间复杂度.

当数量巨大的密文分片存储入数据库时, 数据库中总分片数 N 将会足够大, 由于服务器攻击者没有解密凭证 SDD 代理链接, 攻击者若需要从分片中选择属于同一密文的密钥与密文分片, 也只能采取暴力破解. 此时, 该加密方案 CCA 选择密文攻击[12,13]时间复杂度最好情况下是 $n(n-1)=O(n^2)$, 最坏情况下则是 $n!=O(n^n)$. 而 CCA 选择密文攻击的平均开销是每 n 个分片的读取开销乘以 $(N!+N*(N+1))/2$ 复杂度是 $O(n^n)$. 即在随机预言模型下攻击该方案所需的时间开销无法使用多项式时间计算, 加密方案安全.

5 结论与展望

本文针对我国现在社交软件中带有敏感信息数据泄露的现状, 设计和实现了一种基于微信的数据安全自毁服务平台, 完成了最基本的文件的加密存储以及读取功能, 对该平台的可行性进行了验证. 目前, 该服务平台主要支持图片和 pdf 格式的预览, 如何支持更多格式的文件预览以及进一步丰富服务平台的功能多样性将是下一阶段的研究重点.

参考文献

1 傅晓, 王志坚, 许峰, 等. 一种基于网络的数据自毁方法: 中国, CN102571949A. 2012-07-11.

2 微信公众平台开发者文档. http://mp.weixin.qq.com/wiki/home/index.html.

3 史长江. 微信中的信息安全及防护策略研究. 科技资讯, 2017, 15(14): 10, 12.

4 Fu X, Wang ZJ, Wu H, *et al*. How to send a self-destructing email: A method of self-destructing email system. Proceedings of 2014 IEEE International Congress on Big Data. Anchorage, AK, USA. 2014. 304–309.

5 赵雪梅. AES 加密算法的实现及应用. 常熟理工学院学报, 2010, 24(2): 105–110. [doi: 10.3969/j.issn.1008-2794.2010.02.026]

6 Leach P, Mealling M, Salz R. RFC 4122: A universally unique identifier (UUID) URN namespace. IETF, 2005. [doi: 10.17487/RFC4122]

7 Berners-Lee T, Connolly D. RFC 1866: Hypertext markup language-2.0. ETF, 1995. [doi: 10.17487/RFC1866]

8 Fielding R, Gettys J, Mogul J, *et al*. RFC 2616: Hypertext transfer protocol -- HTTP/1.1. IETF, 1999. [doi: 10.17487/RFC2616]

9 Geambasu R, Kohno T, Levy AA, *et al*. Vanish: Increasing data privacy with self-destructing data. Proceedings of the 18th USENIX Security Symposium. Berkeley, CA, USA. 2009. 299–316.

10 Wu H, Fu X, Wang ZJ, *et al*. Self-destructing data method based on privacy cloud. Proceedings of the International Conference on Logistics, Engineering, Management and Computer Science. Shenyang, China. 2015. 1207–1211. [doi: 10.2991/lemcs-15.2015.241]

11 王晓生, 李莉. 密码学中的随机预言模型与标准模型. 现代电子技术, 2011, 34(17): 98–100, 103. [doi: 10.3969/j.issn.1004-373X.2011.17.029]

12 陈原, 王育民, 肖国镇. 公钥密码体制与选择密文安全性. 西安电子科技大学学报 (自然科学版), 2004, 31(1): 135–139. [doi: 10.3969/j.issn.1001-2400.2004.01.031]

13 梅其祥, 何大可, 唐小虎. 一个加密方案的选择密文安全性的证明. 四川大学学报 (自然科学版), 2006, 43(1): 71–77. [doi: 10.3969/j.issn.0490-6756.2006.01.013]