版权护卫

伴随着互联网和图像技术的快速发展，人们的生活与工作变得越来越方便，交流和娱乐也越来越多样化，但这其中同时也带来了不少安全隐患。回想过去十年的中国互联网暴力野蛮的发展，数字图像的易拷贝和易传播特性，带来了大量的侵权和盗版问题，给著作人的利益带来了严重的损害，对大量的数字行业带来的深刻并且严重的破坏。幸运的是，现在版权问题已经获得越来越多的人重视，社会和国家也在促进版权保护的发展，各个公司更是视版权为财富。即便如此，由于传统的可见水印嵌入方案的版权保护能力已显乏力，不能跟上时代对版权的号召，存在着大量的安全漏洞，可见水印对版权的保护形同虚设。本平台将鲁棒的不可见水印技术和互联网技术进行结合，为此类侵权盗版问题的解决提供了新的视角。虽然不可见水印技术目前仍然存在不足，没有大量的商业化，也不能彻底解决存在的安全问题，但是其提供了一个较为新颖的解决方案，并可以提供比可见水印更强的版权保护。

一、简介

本文产品的名称叫做“版权护卫”，以鲁棒的不可见水印技术为核心，构建了一套精简的版权保护平台，为用户提供最基本的版权保护能力的同时，也能保障图像的观赏性，甚至在图片在一定程度破坏的情况下，同样能够提取出水印。图 1 是本产品的首页。

【图 1】

产品主要提供了“可见水印”、“不可见水印”和“图片注册”的三大功能用以进行版权保护，这三种功能相互独立，每种功能都有其各自的应用场景。1).可见水印。该功能就是现在最场景的水印嵌入方案，嵌入一个肉眼可见的水印，水印内容是一串文本。2).不可见水印。该功能的水印内容同意是一串文本，但是嵌入后不可见，可以通过密钥将嵌入的水印文本提取出来。3).图片注册。该功能将原始图片进行注册，并返回一个注册图片，注册图片和原始图片之间的区别肉眼不可见。用户通过注册图片，可以追溯到该注册图片的注册人，并和注册人进行联系。

二、需求分析

2.1 现状分析

随着数字技术和通信技术的发展，图像成为目前互联网中最受欢迎的多媒体形式之一。2010 年以后，智能手机的大面积普及，也让越来越多的人通过手机拍照来记录生活点滴，再加上手机中各类滤镜的使用与后期手段，手机已经可以拍出不逊色于专业相机的效果，因此人们热于通过移动端的操作来进行图片的制作和分享。

发达的数字图像技术和便捷的分享途径，在给人们带来丰富乐趣和充实的图像内容的同时，也隐藏着多媒体安全问题。在互联网平台上发布自己创作的图片时，容易遭它人盗用，这将会极大侵害图片创作者的个人权益。据国家版权白皮书显示，2016 年全国版权产业整体产值突破 5600 亿元，在侵权盗版方面，网络图片侵权案件排名第二，占到了 24%。再比如，据北京市海淀区法院数据显示，2015 年该法院审理图片著作权案件 1000 多件，2016 年翻倍到 2000 件，2017 年中旬已经有 2800 多件，被告大部分是第三方平台微博、博客、APP 等平台等内容创作和分享平台。中国版权协会驻会副理事长王自强在 2017 年关于图片版权保护的研讨会上说到：“图片行业的地位很尴尬，最新的互联网版权报告竟然就没提图片版权的问题，而实际上，图片的侵权率在众多作品类型中位居第二。”

现阶段，作者在发布图像作品时，为了达到最基本的版权保护功能，通常会往图片中加入可见的水印，以起到保护的作用，水印中通常包含作者或图像本身的相关信息，如图 2 中的照片加入了可见的水印信息“大阪”。

图 2 普通水印

可见水印的引入，改善了图片侵权的现状，也在一定程度上促进了大众的版权保护意识，但是可见水印也存在诸多的缺点亟待解决：

1).安全性过低。数字图像修复技术可以对破损图片进行一定程度的修复，水印本身就是一种可见的图片破损，因此水印极容易被包括数字图像修复技术在内的相关技术进行去除，使得不法分子还原出高质量的原图，进而满足自己的利益。对于一些放置于边边角角的水印而言，直接可以通过截图技术就能去掉水印。

2).对视觉的影响。为了保证安全性，会采用将图像放置图片中央，甚至布满图片的形式，如图 3 所示。这样子的可见水印虽然提高了安全性，不易被截去，但是也对视觉效果带来了较大的破坏，影响了图片的观赏性，不利于图像放在互联网平台上的初衷“传播”。

3).真实性问题。为了提升安全性并且降低对视觉的影响，现在越来越多图像创作者喜欢在图像中引入花式的水印，这些水印看起来较为真实，不容易被攻击者理解为水印，进而被去除，如图 4 所示，碗上的“大阪”标识实际上为水印。虽然这样可以确实能起到效果，但是若被理解为水印，同意也容易被去除，并且容易让观赏者误会。

图 3 强可见水印

图 4 花式可见水印

为了解决可见水印存在的诸多问题，学术界在早期就已经提出了包含 LSB 在内的不可见水印技术，然而目前在不可见水印方面的小程序极少，在小程序搜索栏中搜索“水印”关键词，只能查询出寥寥几款可见水印小程序。不仅仅是小程序，在 App、Web 和桌面应用中，都极少见到不可见水印平台的应用程序。虽然由于技术不够成熟，此类技术持续沉浸于实验室多年，但随着近几年鲁棒不可见水印技术突飞猛进的发展，不可见水印越来越丰满和成熟，已经具备一定的商业使用价值。图 5 中分别显示了原始图像、水印图像和含水印的图像，很明显，从含水印图像中完全无法看出关于水印的信息。

(a) 原始图像　　(b) 水印图像　　(c) 含水印图像

图 5 原始图像、水印图像与含水印图像

2.2 应用场景

2.2.1 版权保护

图片创作者发布不含水印的图片，虽然可以促进图片在互联网中的传播，但是却极易被

盗用和篡改，因此如今很多创作者都嵌入可见水印并发布图片，水印中包含了创作者的相关信息，以便意图使用图片的人可以和图片创作者进行联系，但可见水印对图片带来的视觉影响，以及其较差的安全性，都使得其保护形同虚设。不可见水印技术可以最大程度降低对视觉的影响，其具有的鲁棒性也可以使得图片在一定程度破坏的情况下仍然能够提取水印，并且不可见水印本身也隐藏了图片“含水印”这一事实，避免引起水印攻击者的注意，提升安全性。本产品提供了“不可见水印”功能，提升图片安全性，并且促进图片在互联网中的传播。产品的“不可见水印”支持用户上传原始图片，并输入不可见水印的文本内容，并选填密钥，小程序将会把必要数据上传到服务器，并由服务器嵌入不可见水印，并将含水印图像返回给小程序。意图使用图片的他人，可以在本产品的中输入相关密钥提取出不可见水印，。需要注意的是，在嵌入不可见水印时，图片创作人若是输入了密钥，这一行为意味着图片制作人禁止网络上的他人使用自己的图片，因为除了图片创作者，没有人知道密钥，也就无法提取作者的相关信息并和作者取得联系。图片创作人若是没有输入密钥，意味着每个人都可以提取出不可见水印的信息，水印中往往会有创作人留下的相关信息，意图使用图片的他人便可以和图片制作人取得联系。

我们在进行自己的创意设计时，通常会在网上查阅相关的图片，并引用网上查找到的图片，这些图片往往没有图片创作者的信息，若是在引用时稍不注意，极有可能导致侵权行为。另外一个方面，网络上存在着大量的虚假信息，有人为了谋取一己私利谎称是图片的作者，并且直接采用可见水印或是不可见水印的功能，将会暴露作者的信息，而很多作者为了自己的隐私安全，不愿意对外公布这些信息。通过本产品提供的“图片注册”功能，可以在一定程度上抑制上述问题。图片创作者需要在平台上进行图片的注册，平台将会返回一个注册图片，该图片用于创作者进行发布，普通用户在网上查找到图片时，若该图片是平台的注册图片，则用户可以通过平台查询到图片注册人。很明显，通过这样的方式也可以辨识某人是否为图片作者，也可以避免作者隐私信息遭到泄露。

2.2.2 盗版追踪

数字图像的制作者除了可以嵌入鲁棒的版权水印以保护自身的版权外，还可以嵌入数字指纹以实现盗版追踪。当用户获得作者的认可时，作者将会拷贝图像给该用户，并且该数字图像中包含了唯一标识的水印数据，这样的水印就被称为数字水印。当作者在网上发现自己的数字图像作品被盗版传播时，即可获取作品中的数字水印，以获知是哪位用户进行盗版传播，并绳之以法。产品可以通过“不可见水印”以达到盗版追踪的目的。将数字指纹通过本产品进行不可见水印嵌入，发现盗版后，在本产品中提取数字指纹便可获悉盗版源头。

2.3 目标用户

1).图像创作者

图像创作者在互联网平台上发布图片后，常常希望自己的图片可以得到广泛的传播，又不希望自己的图片被滥用，损害自己的合法利益。图像创作者可以通过平台提供的“可见水印”和“不可见水印”将自己的信息嵌入到图片中，对于不可见水印，用户可以通过平台中的“不可见水印”提取功能，获得图片创作者留下的信息，以便和作者联系。当图片创作者在“不可见水印”功能中，输入了密钥，意味着创作者不愿意别人使用自己的图像，但仍然加入水印，以保证自己的版权信息。

通过“不可见水印”和“可见水印”功能会留下作者的信息，在全网公开后，作者的信息容易被他人获得，造成作者被骚扰等麻烦。为了避免图像创作者的隐私信息遭泄露，平台通过“图片注册”功能，图像创作者可以在平台上注册的自己的创作图，普通用户可以通过图片创作者发布的图片在本平台上向作者发送单一信息，图片创作者视情况和普通用户联系。

本平台为图像创作者提供了第一层的水印上的版权保护，用户希望版权受到最大程度的保护，还需要自己保留原图，并在发表图片前进行著作权的登记，尽可能留下创作证据。平

台后期也会引入不局限于水印的，更全方位的版权保护的服务。

2).希望使用图片的用户

在现今缺乏版权意识的时代，百度和谷歌等搜索引擎会爬到很多没有水印信息的图片，很多互联网用户在分享图片时也较为随意，若不加以注意，对来自互联网的图片的进行滥用，很容易让自己陷入版权纠纷。若是用户在使用前可以和图片创作者取得联系，在引用图片前争取到作者的同意，则能极大的降低此类纠纷的发生。

“可见水印”或者“不可见水印”都是图片创作者将自己的联系方式等信息嵌入到图片中，用户看到作者的信息后可以主动与作者进行联系。为了避免图片创作者的信息随着图片的公开而遭泄露，通过“图片注册-图片追溯-消息发送”的功能，可以有效的避免图片创作者的信息遭到泄露，普通用户通过图片追溯的方式可以查询到作者，并向其发送消息，图片创作者视自身情况考虑是否和用户进行联系。

四、功能详解

4.1 不可见水印

该功能可以往图像中嵌入不可肉眼察觉的水印，需要在选择了原始图片后，输入图片的标题(选填)，不可见水印文字内容(必填)，提取水印的密钥(必填)。由于不可见水印涉及到较为复杂的计算，通过手机来进行运算速度太慢因此当前版本会将图像发送到服务器进行不可见水印内容的嵌入，嵌入完成后会将含密水印返回。图 6 描述了不可见水印嵌入的相关页面。

(a).输入必要信息 (b).等待服务器嵌入完成 (c).服务器返回的图像

【图 6 嵌入不可见水印】

存在水印的嵌入操作，就会有水印的提取操作，需要在选择了图像后，输入水印密钥。图 7 描述了不可见水印提取的相关页面。

【图 7 提取不可见水印】

4.2 图片注册和追溯

图片注册和追溯，是一个闭环的版权保护功能，当图片创作者在本平台上进行了照片的注册后，平台将会绑定图片和作者，并返回一个注册图片(该图片含水印)，作者可以将服务器返回的注册图片用于任何活动的使用。在图 8 给出了图片注册的相关页面。

【图 8 图片注册】

在平台上，用户对注册图片进行溯源，能够找到该图片所对应的注册者，并和该注册者取得联系。在图 9 中给出了图片追溯的相关页面。由于微信是一个较为私人敏感的平台，因此为了避免注册者的隐私遭到泄露，因此追溯成功是不会显示关于注册者任何相关信息的。

【图 9 图片追溯】

4.3 留言收发

对于在平台上注册的图片，用户能够在平台上进行图片追溯，以找到图片的注册者，当找到注册者以后，用户可以向注册者发送消息。为了避免消息发送者的隐私泄露，本平台不会发送关于发送者的任何隐私，因此需要用户在消息中写入自己的愿意透露的别名和联系方式等信息。图 10 中给出了发送留言的相关页面。

【图 10 发送留言】

在留言发送成功后，注册者会接收到未读的留言，注册根据留言发送者中透露的信息，自行选择是否和该用户进行联系。在图 11 中显示了留言查阅的相关页面。当前版本还未引入黑名单功能，在后续会加入。

【图 11 留言查阅】

4.4 可见水印

本产品虽然是以不可见水印技术核心的，但是为了更为平稳的往不可见水印技术进行过度，并且提供更广全方面的版权保护功能，因此同样也提供了可见水印的功能。可见水印的功能较为简单，并不会涉及到复杂的数学计算，因此在小程序端进行完成，并支持离线功能。在图 12 中给出了可见水印的相关页面。

【图 12 可见水印】

4.5 历史记录

考虑到用户进行了水印嵌入操作以后，由于某些原因导致了含水印图像的丢失，因此在服务器上保存了含水印的图像，用户可以通过历史记录查看所有操作的含水印图像，需要注意的是服务器中没有保存用户提供的原图，历史记录中也不会提供原图的记录，并且也不会包含可见水印图像的记录。在图 13 中给出了历史记录的相关页面。

【图 13 历史记录】

五、技术开发方案

5.1 整体架构

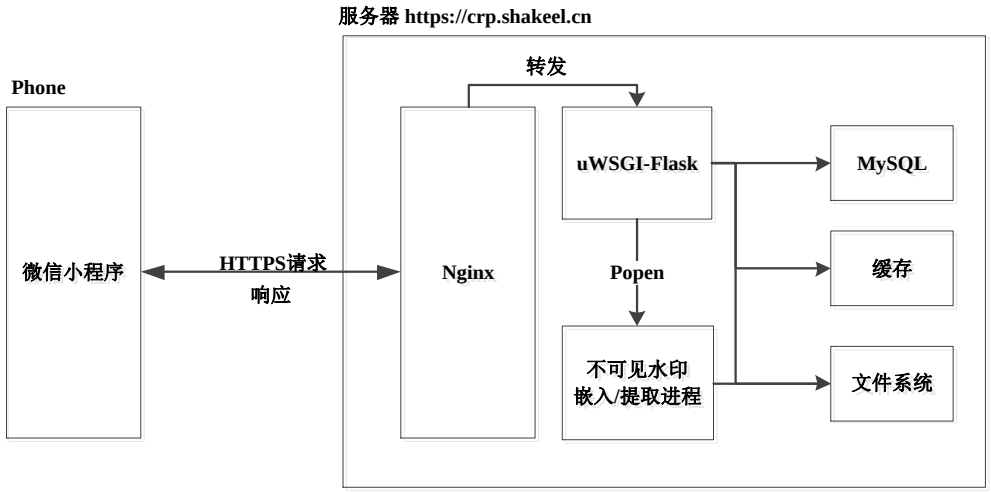


图 1 系统整体框图

5.1.1 服务器端

服务器后台将会提供 HTTPS 接口，以便小程序获得“不可见水印”和“图片注册”的相关服务。后台采用 Python-Flask 进行开发，通过 uWSGI 运行，为了更好的支持 SSL 功能，搭建了 Nginx 服务器接收 HTTPS 请求，并将请求转发给 uWSGI 以处理。

Python 后台在产品中的主要作用是接收和返回网络 IO 任务以及相关数据的出库入库，而对于不可见水印的嵌入和提取过程，这是一个 CPU 密集的任务，考虑到 C++在 CPU 密集型任务上处理的优势，因此对于不可见水印的嵌入和提取请求，Python 将会转交给 C++进程进行处理。目前转交给 C++处理的形式较为简单和粗暴，是通过 Python 调用 C++进程的形式进行的，这样的形式缺点主要在于对于高并发情况下，大量的进程创建和销毁，这将会成为性能瓶颈，因此后期考虑通过 C++进程进行端口监听，Python 通过 Protobuf 将任务序列化，并交给 C++进程，C++进程接收到 Python 的任务后，提交任务给线程池来处理不可见水印提取和嵌入的任务，如图所示。改进的机制不但避免了大量的进程创建销毁所需要花费的代价，并且限制了进程的无限制创建，通过线程池机制可以在高并发环境下削峰填谷。

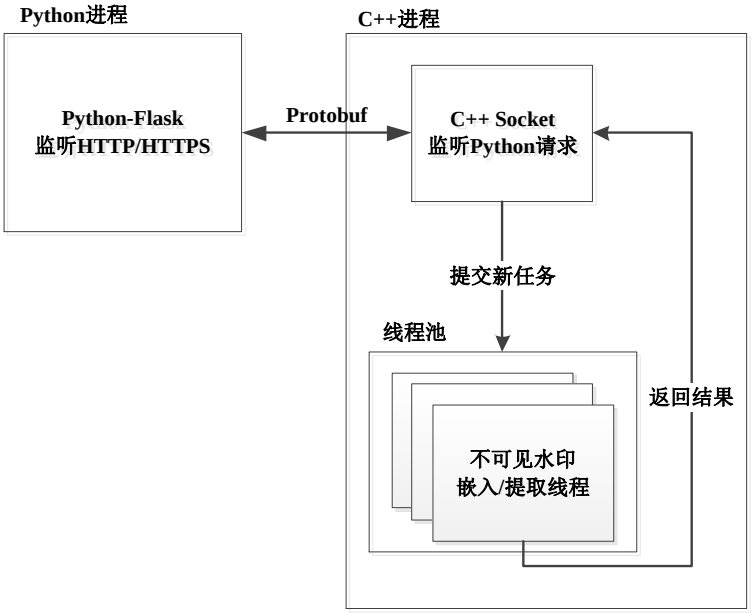


图 2 改进的 Python 转发 C++机制

5.1.2 小程序端

5.2 解决方案

5.2.1 小程序端

5.2.2 服务器端

1). 会话机制

Session(会话)是 Web 架构中常见的机制，通过在 cookie 中保存 sessionId 的形式来实现，用来在服务器上保存当前会话的重要数据，例如登录信息和配置信息等。在微信小程序中，并没有实现 Cookie 机制，进而无法支持 Web 的 Session 机制，因此需要我们自己设计 Session 机制。当小程序每次登录的时候，都会发送一个会话建立请求到服务器，该请求中包含了微信登录凭证 code 和设备唯一码 did，便于服务器获取建立会话并获取该会话所对应的微信 ID。服务器在会话建立的时候，会立即将微信 ID 放到会话的缓存中，以便获取后续每个请求所对应微信 ID。

同一个设备上的同一个微信用户，若已经和服务器建立了会话后，由于某些原因(例如 sessionId 丢失)将会再次尝试和服务器建立会话，服务器将会进行会话合并，即返回该服务器和该微信小程序已有会话的 sessionId。微信自身虽然可以避免同一个微信用户在不同的设备上进行登录，但由于本产品是基于 HTTPS 的，因此极易被模拟登入，会有同一个微信用户在不同的设备上登录本平台的风险，进而导致同一个会话服务于多个设备(会话合并造成的)，因此需要设备唯一码 did 进行标识。微信小程序本身无法获取 did，因此需要到服务器上获取 did。服务器的 did 接口每次调用都会返回一个不一样的 ID 号，小程序在第一次登入的时候将会缓存该 ID 号，作为自己的设备唯一码，以后每次会话建立请求都应该带上该设备唯一码。图 x 是服务器接收到会话建立请求后的处理流程图。

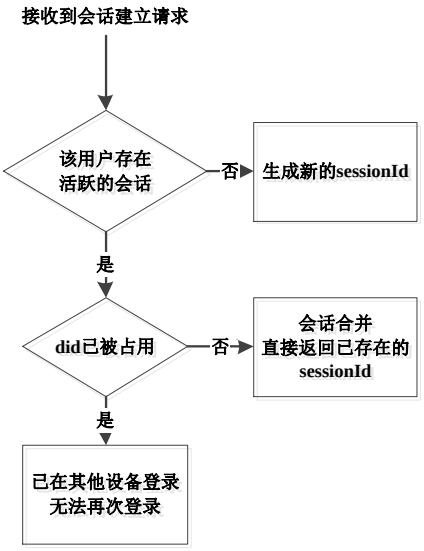


图 3 会话建立流程图

服务器有两种方式销毁接口，一种是服务器提供了会话销毁接口，在小程序退出的时候应该调用会话销毁接口。另一种时超时自动销毁，会话每次建立 1 小时候，服务器会自动销毁掉会话，会话合并将会重新刷新持续时间为 1 小时。持续时间是为了避免小程序端由于某些原因，导致会话销毁请求无法发送出去，进而占用服务器资源。小程序端应该持续检测用户的活动情况，当用户存在活跃点击的时候，小程序将会周期性发送会话建立请求，触发会话合并，进而方便刷新会话超时时间。图 x 是整个会话的生命周期时序图。

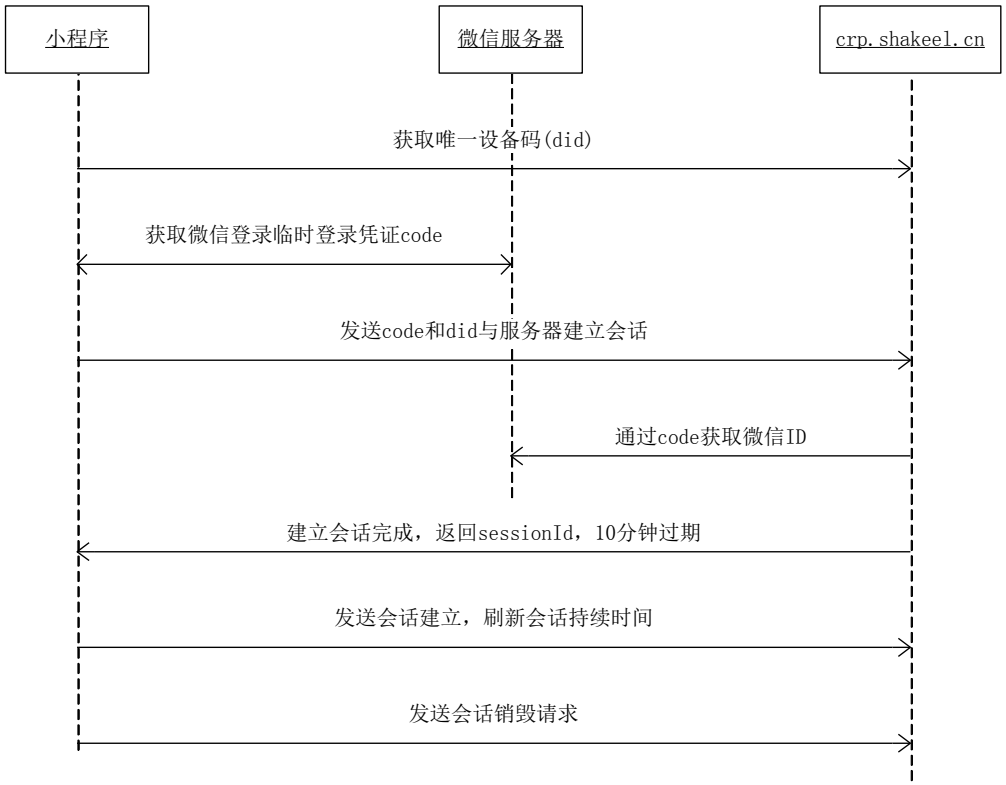


图 4 会话生命周期

2). 不可见水印嵌入机制

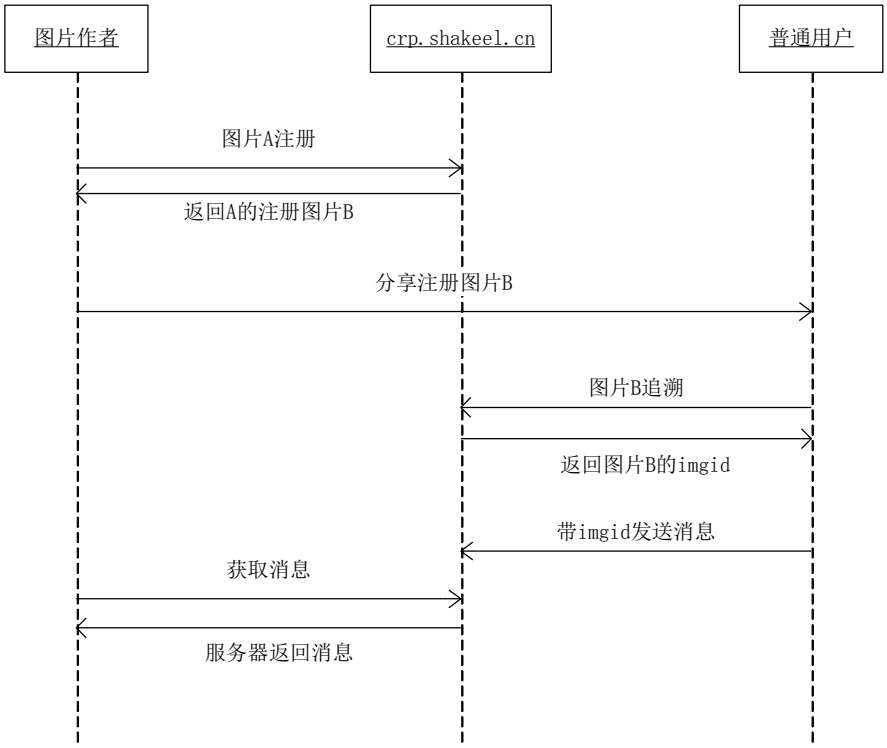
一张图片的大小有限，而嵌入的数据无限制增大时，将会严重影响图像质量以及增大图片文件的大小。“不可见水印”为了避免对图像质量影响较大或是增大图像二进制数据大小，

因此采用了隐藏 *imgnum* 的方案，将 md5(*imgnum*)和不可见水印的内容绑定在一起。

imgnum 是一个 64 比特的数据，并且前 16 比特为校验位，用于标志该 *imgnum* 是否为平台所生成的不可见水印，而 md5(*imgnum*)是图像的 ID 号，即 *imgid*。在嵌入不可见水印时，为了避免图片已经在平台上进行过水印嵌入或是图片注册，将会首先提取 *imgnum*，首先检查校验位，校验不通过则认为该图片不含本平台的水印，可以进行嵌入。若校验通过，再检查数据库中是否存在该 *imgid*， 若存在则认为该图像已经在图片嵌入过数据，不能再次进行嵌入。不可见水印在提取时，在提取 *imgnum* 后，也需要先检查校验位，校验不同则在平台上没有因此数据，若校验通过则到数据库中去寻找 *imgid*，若没有找到该同样认为该图像没有因此数据，否则返回图像不可见水印。

3). 图片追溯的隐私保护机制

图片注册相比于不可见水印的最大的优点是在保护版权的同时，避免图片创作者在图片中加入自己的信息，导致的隐私泄露。图片追溯可以让意图使用图片的人在平台上和作者取得联系，作者视情况自行回复，避免了隐私的泄露。因此在整个图片注册，图片追溯和消息发送的流程中，都不应该泄露除了图片本身以外的任何有关作者的信息。



【图 x 图片追溯时序图】

在图 x 中给出整个流程时序图，首先图片注册就会先生成一个 *imgid*，将其作为水印信息嵌入到图片中返回给小程序，并将 *imgid* 和微信 ID 在数据库中绑定在一起。图片追溯时，后台将会提取出水印作为 *imgid*，并且当 *imgid* 在数据库中存在时，则认为图片经过了注册，能够追溯到原作者，这时将会返回小程序该图片的 *imgid*。用户在追溯成功后，可以发送消息，消息发送会带上追溯成功时返回的 *imgid*，服务器接收到了消息发送的请求，会将 *imgid* 所对应的微信用户查询出来，再将消息发送给该微信用户。很明显，整个追溯过程都不会泄漏关于图片注册的任何信息，都是由希望使用图片的用户发起的所有操作，并等待图片作者的回复，这样确保了图片作者的隐私不被泄露。

4). 变分辨率问题

平台在进行图片注册或是嵌入不可见水印时，会在保持宽高比的条件下，对分辨率进行变更。一次是在小程序发送图片注册或嵌入不可见水印请求时，对上传上来的图片进行分辨

率变更， 另一次是算法进程生成了含水印图像时，会额外生成一张小分辨率的图片。

在图片注册或嵌入不可见水印时，会将图片变分辨率主要是基于两点考虑：a).水印鲁棒性问题。实际嵌入的数据为了达到一定的鲁棒性会进行扩频，图像分辨率越大，将会有更大的扩频空间，以增加鲁棒性。b).微信转发图片对图像分辨率的变更。经过反复的测试发现，微信在发送图片时，会以一些规律对图片在保持宽高比的情况下进行分辨率的变更。含水印图片在变更分辨率后鲁棒性会变差，为了避免这样的情况出现，平台将会首先对图片分辨率进行变更，避免微信转发图片时对图片分辨率的变更导致提取水印时的困难。

当小程序发送获取消息和获取历史记录的请求时，为了让用户知道历史记录对应的图片以及消息提醒所对应的图片，服务器将会返回对应图片的 URL。因为含水印的图片都是非常大的，不但影响传输和渲染效率，也增大小程序的缓存时的容量压力。由于移动端设备童通常较小，历史记录和消息提醒的图片预览并不需要获得原图，因此在图片注册和嵌入不可见水印以后，都会生成小分辨率图像，方便历史记录和消息提醒中的预览。后期考虑为了适配更多种类的移动端设备，例如平板，大型号手机等，将会引多分辨率的方案，并由小程序端指定需要哪种分辨率的图片，以达到自适应设备的功能。

5.3 核心算法

六、团队的组成与分工

1).陈智隆(队长)

负责鲁棒水印核心算法编写，包括了：1).测试微信发送图片对图片本身的 JPEG 压缩和变分辨率情况。2).查阅相关文献，进行鲁棒水印的 Matlab 仿真，并测试在微信发送后水印的提取情况。3).采用 OpenCV 库，将鲁棒水印算法从 Matlab 移植到 C++平台，并分别编译了 Windows 和 Linux 版本的动态链接库，以便 Python 调用。

2).李素静

负责小程序端的界面显示以及所有交互效果，包括了：1).了解并熟悉小程序端开发组件和工具。2).设计并开发了小程序的显示界面。3).完成人机交互效果，优化用户体验。

3).卢帅吉

负责系统架构的设计以及系统的开发与部署，包括了：1).通过腾讯云架设服务器运行环境，如域名备案，Nginx 搭建，Python-Flask 环境搭建，SSL 注册等。2).设计服务器的 HTTPS 接口。3).开发服务器 CRUD 业务代码。