登录注册功能相关主题

一、密码

用户名和密码如果明文传输，会导致安全问题，一旦密码泄露，数据就有可能会被盗取，所以需要加密传输。

可逆加密算法：加密后可以解密得到密码原文

对称加密：加密、解密用相同的秘钥。
优点是算法公开、加密速度快、效率高；缺点是不安全
使用场景：保存手机号等需解密的信息
常见算法： AES、DES、RC4、RC5、RC6
非对称加密：加密使用公钥，解密使用私钥。私钥签名，公钥可验证是否被篡改
优点是安全性高；缺点是加密解密时间长、速度慢
使用场景：用于签名或认证
常见算法：RSA、DSA
不可逆加密算法：加密后不能反向解密
使用场景：储存用户敏感信息，比如密码、银行账号
常见算法：MD5、SHA

常见的 7 种密码加密方式

以下几种方式是常见的密码保存方式：

算法	特点	有效破解方式	破解难度	其它
对称加密	可以解密出明文	获取密钥	中	需要确保密钥不泄露
单向 HASH	不可解密	碰撞、彩虹表	中	
特殊 HASH	不可解密	碰撞、彩虹表	中	需要确保“盐”不泄露
Pbkdf2	不可解密	无	难	需要设定合理的参数
BCrypt	不可解密	无	难	需要设定合理的参数
SCrypt	不可解密	无	难	需要设定合理的参数
Argon2	不可解密	无	难+	

1、使用对称加密算法来保存

使用情况： ★★☆☆☆

比如 3DES、AES 等算法，使用这种方式加密是可以通过解密来还原出原始密码的，当然前提条件是需要获取到密钥。不过既然大量的用户信息已经泄露了，密钥很可能也会泄露，当然可以将一般数据和密钥分开存储、分开管理，但要完全保护好密钥也是一件非常复杂的事情，所以这种方式并不是很好的方式。

2、使用 MD5、SHA1 等单向 HASH 算法保护密码

使用情况： ★★★☆☆

使用这些算法后，无法通过计算还原出原始密码，而且实现比较简单，因此很多互联网公司都采用这种方式保存用户密码，曾经这种方式也是比较安全的方式，但随着彩虹表技术的兴起，可以建立彩虹表进行查表破解，目前这种方式已经很不安全了。

3、特殊的单向 HASH 算法

使用情况： ★★★☆☆

由于单向 HASH 算法在保护密码方面不再安全，于是有些公司在单向 HASH 算法基础上进行了加盐、多次 HASH 等扩展，这些方式可以在一定程度上增加破解难度，对于加了“固定盐”的 HASH 算法，需要保护“盐”不能泄露，这就会遇到“保护对称密钥”一样的问题，一旦“盐”泄露，根据“盐”重新建立彩虹表可以进行破解，对于多次 HASH，也只是增加了破解的时间，并没有本质上的提升。

4、PBKDF2

使用情况： ★★★★☆

该算法原理大致相当于在 HASH 算法基础上增加随机盐，并进行多次 HASH 运算，随机盐使得彩虹表的建表难度大幅增加，而多次 HASH 也使得建表和破

解的难度都大幅增加。使用 PBKDF2 算法时，HASH 算法一般选用 sha1 或者 sha256，随机盐的长度一般不能少于 8 字节，HASH 次数至少也要 1000 次，这样安全性才足够高。一次密码验证过程进行 1000 次 HASH 运算，对服务器来说可能只需要 1ms，但对于破解者来说计算成本增加了 1000 倍，而至少 8 字节随机盐，更是把建表难度提升了 N 个数量级，使得大批量的破解密码几乎不可行，该算法也是美国国家标准与技术研究院推荐使用的算法。

PBKDF2 已经存在很长时间了，像之前文章讨论的一样，它有点过时了：轻松的在多核系统（GPU）上实现并行，这对于定制系统（FPGA/ASIC）来说微不足道。

5、BCrypt

使用情况： ★★★★☆

BCrypt 在 1999 年就产生了，并且在对抗 GPU/ASIC 方面要优于 PBKDF2，但是我还是不建议你在新系统中使用它，因为它在离线破解的威胁模型分析中表现并不突出。 尽管有一些数字加密货币依赖于它（即：NUD），但它并没有因此获得较大的普及，因此，FPGA/ASIC 社区也并没有足够的兴趣来构建它的硬件实现。 话虽如此，Solar Designer（OpenWall）、Malvoni 和 Knezovic（萨格勒布大学）在 2014 年撰写了一篇论文，这篇文章描述了一种混合使用 ARM/FPGA 的单片系统来攻击该算法。

6、SCrypt

使用情况： ★★★★☆

SCrypt 在如今是一个更好的选择：比 BCrypt 设计得更好（尤其是关于内存方面）并且已经在该领域工作了 10 年。另一方面，它也被用于许多加密货

币，并且我们有一些硬件（包括 FPGA 和 ASIC）能实现它。 尽管它们专门用于采矿，也可以将其重新用于破解。

7、Argon2

使用情况： ★★★★★

Argon2 基于 AES 实现，现代的 x64 和 ARM 处理器已经在指令集扩展中实现了它，从而大大缩小了普通系统和攻击者系统之间的性能差距，

Argon2 有三个主要的版本：

- Argon2i 是对抗侧信道攻击的最安全选择，Argon2i 使用与数据无关的内存访问，这是密码哈希的首选方法，Argon2i 对内存进行了更多的传递，以防止权衡攻击的发生。
- Argon2d 是抵抗 GPU 破解攻击的最安全选择，并且 Argon2 在 2015 年 7 月赢得了密码哈希竞赛。Argon2d 使用依赖数据的内存访问，这使得它很适合用于加密数字货币和工作量证明的应用程序，而不会受到侧信道定时攻击的威胁。
- Argon2id 在内存第一次迭代的前半部分充当 Argon2i，其余部分则充当 Argon2d。因此，基于时间、 空间的平衡，它既提供了侧信道攻击保护也节约了暴力开销。

如果你担心侧信道攻击（例如：恶意数据缓存加载/幽灵漏洞，它允许通过基于缓存的侧信道读取同一硬件上其他正在运行的进程的私有内存数据），你应该使用 Argon2i，否则使用 Argon2d。 如果你不确定或你对混合方法感到满意，你可以使用 Argon2id 来获得两个方面的优势。

源代码可以在 Github 上获得，使用兼容 C89 的 C 语言编写，并在知识共享许可协议下获取许可，并且可以在大多数 ARM、x86 和 x64 架构的硬件上编译。

…

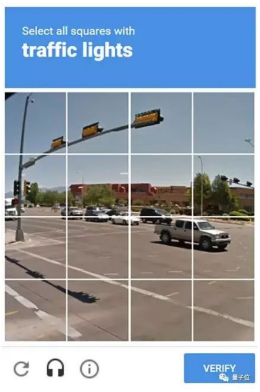
温馨提醒：

在 2019 年之后，就有相关专家提出建议尽量不要使用 PBKDF2 或 BCrypt，并强烈建议将 Argon2（最好是 Argon2id）用于最新系统。而 Scrypt 是当 Argon2 不可用时的不二选择，但要记住，它在侧侧信道泄露方面也存在相同的问题。

二、验证码

又被称作全自动区分计算机和人类的图灵测试（Completely Automated Public Turing test to tell Computers and Humans Apart，简称 CAPTCHA），俗称验证码，是一种区分用户是计算机和人的公共全自动程序。验证码的主要目的是强制人机交互来抵御机器自动化攻击，为了确保服务器系统的稳定和用户信息的安全，越来越多的网站采用了验证码技术。

验证码的形式：



破解方法

复杂的 Captcha 对文字识别，图形图像处理以及人工智能专家来说都是一个很大的挑战，但是这并不能阻止互联网上那些 Bot 创造者们的脚步，一些新兴的破解 Captcha 的办法也就应运而生，最常用的包括两种方法：

1. 利用现成的网站（很多是吸引眼球的不正当网站）的高流量，让那些过路者免费帮忙输入验证码。

2. 直接付费利用人力资源输入验证码，就是一个专门进行此类勾当的网站。

当然 Captcha 并不是唯一的解决互联网垃圾的手段，相信以后会有更多的更方便更安全的技术出现在人们的眼前。

参考：

https://www.php.cn/faq/493422.html

https://github.com/xitu/gold-miner/blob/master/TODO1/password-hashing-pbkdf2-scrypt-bcrypt-and-argon2.md

https://blog.csdn.net/sinat_34626741/article/details/124107868

https://zhuanlan.zhihu.com/p/113984411

https://zhuanlan.zhihu.com/p/571500021

https://zhuanlan.zhihu.com/p/398514325