<https://mp.weixin.qq.com/s/F5r6D_zW8XdbdMvxe3k6Kg>

# 用户密码加密存储十问十答，一文说透密码安全存储

### 我们数据库的权限管理十分严格，敏感信息开发工程师都看不到，密码明文存储不行吗？

不行。存储在数据库的数据面临很多威胁，有应用程序层面、数据库层面的、操作系统层面的、机房层面的、员工层面的，想做到百分百不被黑客窃取，非常困难。

如果密码是加密之后再存储，那么即便被拖库，黑客也难以获取用户的明文密码。可以说，密码加密存储是用户账户系统的底裤，它的重要性，相当于你独自出远门时缝在内衣里的钱，虽然你用到他们的概率不大，但关键时刻他们能救命。

### 那用加密算法比如AES，把密码加密下再存，需要明文的时候我再解密。

不行。这涉及到怎么保存用来加密解密的密钥，虽然密钥一般跟用户信息分开存储，且业界也有一些成熟的、基于软件或硬件的密钥存储方案。但跟用户信息的保存一样，想要密钥百分百不泄露，不可能做到。用这种方式加密密码，能够降低黑客获取明文密码的概率。但密钥一旦泄露，用户的明文密码也就泄露了，不是一个好方法。

另外，用户账户系统不应该保存用户的明文密码，在用户忘记密码的时候，提供重置密码的功能而不是找回密码。

### 保存所有密码的HASH值，比如MD5。是不是就可以了？

不是所有的HASH算法都可以，准确讲应该是Cryptographic Hash。Cryptographic Hash具有如下几个特点：

* 给定任意大小任意类型的输入，计算hash非常快；
* 给定一个hash，没有办法计算得出该hash所对应的输入；
* 对输入做很小改动，hash就会发生很大变化;
* 没有办法计算得到两个hash相同的输入;

虽然不是为加密密码而设计，但其第2、3、4三个特性使得Cryptographic Hash非常适合用来加密用户密码。常见的Cryptographic Hash有MD5、SHA-1、SHA-2、SHA-3/Keccak、BLAKE2。

从1976年开始，业界开始使用Cryptographic Hash加密用户密码，最早见于Unix Crypt。但MD5、SHA-1已被破解，不适合再用来保存密码。

### 那我保存用户密码的SHA256值。

不行。黑客可以用查询表或彩虹表来破解用户密码。注意是破解密码不是破解sha256，能根据sha256破解密码的原因是，用户密码往往需要大脑记忆、手工输入，所以不会太复杂，往往具有有限的长度、确定的取值空间。

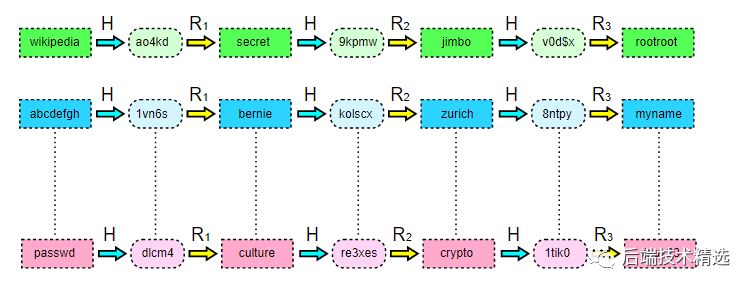
短的取值简单的密码可以用查询表破解

比如8位数字密码，一共只有10^8=100000000种可能。一亿条数据并不算多，黑客可以提前吧0-99999999的sha256都计算好，并以sha256做key密码为value存储为一个查询表，当给定sha256需要破解时，从表中查询即可。

取值相对复杂，且长度较长的密码，可以用彩虹表破解

比如10位，允许数字、字母大小写的密码，一共有(10+26+26)^10~=84亿亿种可能，记录非常之多难以用查询表全部保存起来。这时候黑客会用一种叫做彩虹表的技术来破解，彩虹表用了典型的计算机世界里解决问题的思路，时间空间妥协。

在这个例子里面，空间不够，那就多花一些时间。在彩虹表中，可以将全部的sha256值转化为长度相同的若干条hash链，只保存hash链的头和尾，在破解的时候先查询得到sha256存在于哪条hash链中，然后计算这一条hash链上的所有sha256，通过实时比对来破解用户密码。



上图图展示了一个hash链长度为3的彩虹表，因为在hash链中需要将hash值使用R函数映射回密码取值空间，为了降低R函数的冲突概率，长度为K的hash链中，彩虹表会使用k个R函数，因为每次迭代映射回密码空间使用的R函数不一样，这种破解方法被称作彩虹表攻击。

实际的情况Hash链要比远比上例更长，比如我们的例子中全部的84亿亿个sha256存不下，可以转化为840亿条长度为1千万的sha链。对彩虹表原理感兴趣的话，可以阅读它的维基百科。

网路上甚至有一些已经计算好的彩虹表可以直接使用，所以直接保存用户密码的sha256是非常不安全的。

### 怎样避免彩虹表攻击？

简单讲，就是加盐。一般来讲用户密码是个字符串key、盐是我们生成的字符串salt。原来我们保存的是key的hash值HASH(key)，现在我们保存key和salt拼接在一起的hash值HASH(key+salt)。

这样黑客提前计算生成的彩虹表，就全都失效了。

### 盐应该怎么生成，随机生成一个字符串？

这是个好问题，并不是加个盐就安全了，盐的生成有很多讲究。

**使用CSPRNG（Cryptographically Secure Pseudo-Random Number Generator）生成盐，而不是普通的随机数算法；**

CSPRNG跟普通的随机数生成算法，比如C语言标准库里面的rand()方法，有很大不同。正如它的名字所揭示，CSPRNG是加密安全的，这意味着用它产生的随机数更加随机，且不可预测。常见编程语言都提供了CSPRNG，如下表：



**盐不能太短**

想想查询表和彩虹表的原理，如果盐很短，那意味着密码+盐组成的字符串的长度和取值空间都有限。黑客完全可以为密码+盐的所有组合建立彩虹表。

**盐不能重复使用**

如果所有用户的密码都使用同一个盐进行加密。那么不管盐有多复杂、多大的长度，黑客都可以很容易的使用这个固定盐重新建立彩虹表，破解你的所有用户的密码。如果你说，我可以把固定盐存起来，不让别人知道啊，那么你应该重新读一下我关于为什么使用AES加密不够安全的回答。

即便你为每一个用户生成一个随机盐，安全性仍然不够，因为这个盐在用户修改密码时重复使用了。应当在每一次需要保存新的密码时，都生成一个新的盐，并跟加密后的hash值保存在一起。

注意：有些系统用一个每个用户都不同的字段，uid、手机号、或者别的什么，来作为盐加密密码。这不是一个好主意，这几乎违背了上面全部三条盐的生成规则。

### 那我自己设计一个黑客不知道的HASH算法，这样你的那些破解方法就都失效了。

不可以。

首先如果你不是一个密码学专家，你很难设计出一个安全的hash算法。不服气的话，你可以再看一遍上面我关于Cryptographic Hash的描述，然后想一想自己怎么设计一个算法可以满足它的全部四种特性。就算你是基于已有的Cryptographic Hash的基础上去设计，设计完之后，也难以保证新算法仍然满足Cryptographic Hash的要求。而一旦你的算法不满足安全要求，那么你给了黑客更多更容易破解用户密码的方法。

即便你能设计出一个别人不知道的Cryptographic Hash算法，你也不能保证黑客永远都不知道你的算法。黑客往往都有能力访问你的代码，想想柯克霍夫原则或者香农公里：

密码系统应该就算被所有人知道系统的运作步骤，仍然是安全的。

### 为每一个密码都加上不同的高质量的盐，做HASH，然后保存。这样可以了吧？

以前是可以的，现在不行了。计算机硬件飞速发展，一个现代通用CPU能以每月数百万次的速度计算sha256，而GPU集群计算sha256，更是可以达到每秒10亿次以上。这使得暴力破解密码成为可能，黑客不再依赖查询表或彩虹表，而是使用定制过的硬件和专用算法，直接计算每一种可能，实时破解用户密码。

那怎么办呢？回想上面关于Cryptographic Hash特性的描述，其中第一条：

给定任意大小任意类型的输入，计算hash非常快

Cryptographic Hash并不是为了加密密码而设计的，它计算非常快的这个特性，在其他应用场景中非常有用，而在现在的计算机硬件条件下，用来加密密码就显得不合适了。针对这一点，密码学家们设计了PBKDF2、BCRYPT、SCRYPT等用来加密密码的Hash算法，称作Password Hash。在他们的算法内部，通常都需要计算Cryptographic Hash很多次，从而减慢Hash的计算速度，增大黑客暴力破解的成本。可以说Password Hash有一条设计原则，就是计算过程能够按要求变慢，并且不容易被硬件加速。

### 应该使用哪一种Password Hash？

PBKDF2、BCRYPT、SCRYPT曾经是最常用的三种密码Hash算法，至于哪种算法最好，多年以来密码学家们并无定论。但可以确定的是，这三种算法都不完美，各有缺点。其中PBKDF2因为计算过程需要内存少所以可被GPU/ASIC加速，BCRYPT不支持内存占用调整且容易被FPGA加速，而SCRYPT不支持单独调整内存或计算时间占用且可能被ASIC加速并有被旁路攻击的可能。

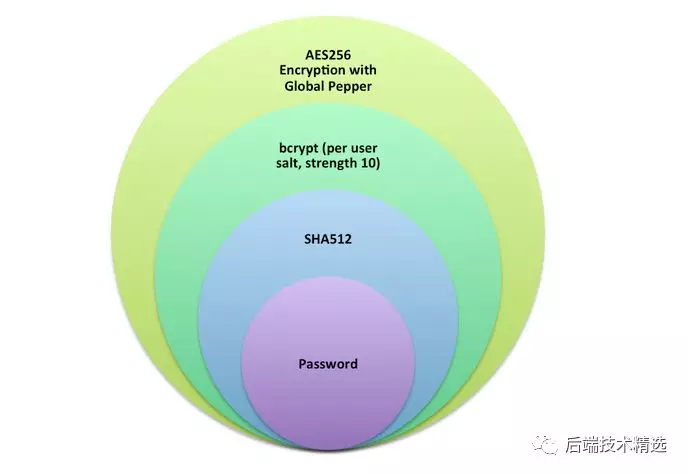
2013年NIST（美国国家标准与技术研究院）邀请了一些密码学家一起，举办了密码hash算法大赛（Password Hashing Competition），意在寻找一种标准的用来加密密码的hash算法，并借此在业界宣传加密存储用户密码的重要性。大赛列出了参赛算法可能面临的攻击手段：

* 加密算法破解（原值还原、哈希碰撞等，即应满足Cryptographic Hash的第2、3、4条特性）;
* 查询表/彩虹表攻击；
* CPU优化攻击；
* GPU、FPGA、ASIC等专用硬件攻击；
* 旁路攻击;

最终在2015年7月，Argon2算法赢得了这项竞赛，被NIST认定为最好的密码hash算法。不过因为算法过新，目前还没听说哪家大公司在用Argon2做密码加密。

### 一路问过来好累，能不能给我举个例子，大公司是怎么加密用户密码的？

今年（2016）Dropbox曾发生部分用户密码数据泄露事件，当时其CTO表示他们对自己加密密码的方式很有信心，请用户放心。随后，Dropbox在其官方技术博客发表名为《How Dropbox securely stores your passwords》的文章，讲述了他们的用户密码加密存储方案。



如上图所示，Dropbox首先对用户密码做了一次sha512哈希将密码转化为64个字节，然后对sha512的结果使用Bcrypt算法（每个用户独立的盐、强度为10）计算，最后使用AES算法和全局唯一的密钥将Bcrypt算法的计算结果加密并保存。

**博文中，Dropbox描述了这三层加密的原因：**

首先使用sha512，将用户密码归一化为64字节hash值。因为两个原因：一个是Bcrypt算法对输入敏感，如果用户输入的密码较长，可能导致Bcrypt计算过慢从而影响响应时间；另一个是有些Bcrypt算法的实现会将长输入直接截断为72字节，从信息论的角度讲，这导致用户信息的熵变小；

然后使用Bcrypt算法。选择Bcrypt的原因，是Dropbox的工程师对这个算法更熟悉调优更有经验，参数选择的标准，是Dropbox的线上API服务器可以在100ms左右的时间可计算出结果。另外，关于Bcrypt和Scrypt哪个算法更优，密码学家也没有定论。同时，Dropbox也在关注密码hash算法新秀Argon2，并表示会在合适的时机引入；

最后使用AES加密。因为Bcrypt不是完美的算法，所以Dropbox使用AES和全局密钥进一步降低密码被破解的风险，为了防止密钥泄露，Dropbox采用了专用的密钥保存硬件。Dropbox还提到了最后使用AES加密的另一个好处，即密钥可定时更换，以降低用户信息/密钥泄露带来的风险。