**密码加密方法探究**

1. **案例描述**

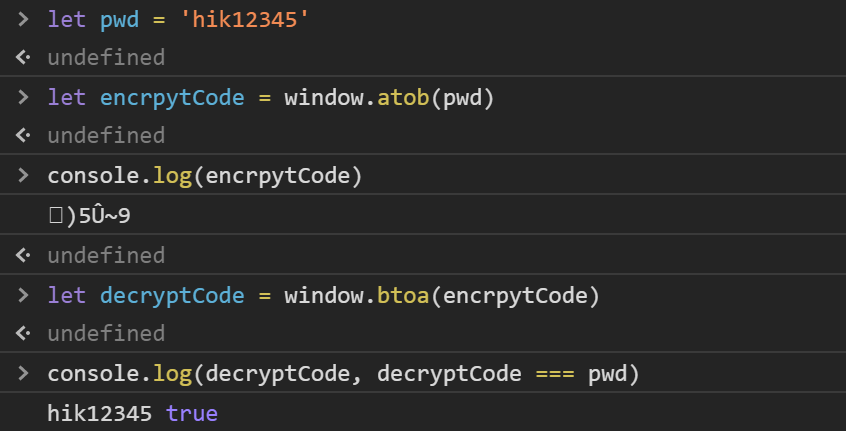
在前端需要密码登录的业务场景中，用户在登录时，通常需要对输入的密码进行加密。对此有多种加密方式可以实现，本文通过对比几种加密方法，并分析前后端加密流程，尝试找出一种安全性较高且易于实现的加密方法。

1. **案例分析与解决**

比较常见的前端对密码加密可以采用Base64，MD5，SHA-1, SHA-2这几种方法，而在具体实现中可能会用到二次加密和加盐等操作。本文对各种方法的加密原理做出简要说明，并分析比较各种方法面临的风险，比较各种方法的优劣。

1. **加密方法的分析与比较**
   1. **Base64加密**

这种加密方式对于前端来说不需要引入第三方库，通过window.atob() 与window.btoa()方法可以实现对Base64格式的数据进行解码和编码，其兼容性是主流浏览器，IE10及以上。但问题在于通过window.atob()加密的密文，是可以通过window.btoa()进行解码的。而我们根本不希望密文可以被解码，后端从前端拿到密文并不需要还原成明文。在用户登录的时候，前端将用算法加密用户输入的密码， 然后将密码对应的密文发送到后端。 后端将用户输入的密码密文，与数据库中存储的用户密码密文（用户首次注册时保存）比对，如果一致则判定登录成功，不一致则失败。这样既保证了密码除了用户自身不被任何人知道，也保证可以有效判定用户是否输入了正确的密码。通过Base64加密和解密的方式如下图所示。



此种方式只比明文传输密码好一些，但对于密码加密来说，Base64完全不是一种理想的加密方法。因为其加密的密文可以通过解密算法还原成密码原文。

* 1. **MD5加密**

MD5加密是一种常见的加密方式，前端采取此种方式加密时，需要引入MD5加密算法的第三方库。MD5加密方式是本质上是一种不可逆的哈希算法，无论输入是什么，输出的都是一个128bit的值，如果用16进制表示，即是一个32位字符串，还有一种MD5加密出的密文为16位字符串，其实是在加密的32位字符串中去掉前八位和后八位，留下中间的16位密文得到的。MD5不可逆的原因是因为它是一种散列函数, 在计算过程中原文的信息是有损失的。因为输入的明文有无限种情况，而输出是一个128bit一个只能表示有限多种情况的值。所以也可以把经过MD5加密的密文理解为明文的一个数字签名，尽管MD5加密算法是公开的，但是密文不可能通过算法还原成明文。

密文不能通过算法解密成为明文，是不是就说明绝对安全了呢。其实攻击者可以通过撞库的方式暴力破解，面对一个MD5加密的密文，也可以用查表法，其实就是黑客事先准备了一个字典，存储了尽可能多的原文和在MD5算法下对应的哈希值。通过反向查表的方式获得对应密文的明文，特别是字典的数据已经累计到一定程度之后，破解的可能性就变得更高。而对于我们的密码来说，密码一般都只有十几位上下，所以一般建议我们的密码不要过短，而且不要是某个英文单词，或者存在某种明显的规律，最好是数字字母和特殊字符的随机组合。因为攻击者所用的字典虽然不可能包含所有的明文组合，因为明文组合是无限的，但却可以有针对性的收纳了在一定长度之内，很多有规律的组合, 这样就大大增加被破解的可能性。

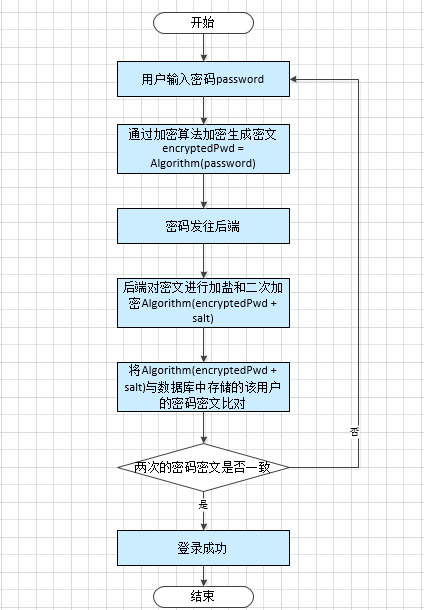
除了引导用户保证密码位数和无规律性， 对于开发者而言，给密码加盐就是一种很好的提高安全性的手段，给攻击者增加一定的破解负担。盐指的是人为给被密码添加无规律的字符串以保证更高的安全性，加盐的方式后续会有更详细的说明。

* 1. **SHA-1和SHA-2**

SHA-1同样是一种哈希加密算法， 由美国国家标准技术研究院（NIST）与美国国家安全局（NSA）共同设计，曾经一些重要的场合都选择SHA-1来做数字签名。相比MD5安全性更高一些。 但虽然MD5和SHA-1曾经是使用最广泛的摘要算法，在2004年国际密码学会议，中国山东大学的王小云教授破解了MD5加密算法， 而很快在2005年SHA-1加密也被王小云教授破解。因此MD5，SHA-1加密在安全性要求较高的场合也逐渐被取代。而一个重要的更安全的替代加密算法即是SHA-2，SHA-2家族包括SHA-224, SHA-256, SHA-384和SHA-512，它们生成的摘要密文分别就是224, 256, 384, 512位，到目前为止还没有SHA-2破解的报告。出于安全性的考虑，采用SHA-2加密是一种更优的选择。

* 1. **加盐与前后端加密**

一种安全性较高的加密方式是前后端都对密码进行加密，前端加密的主要考虑是，避免密码在网络中明文传输，且防止后端开发人员能够拿到明文密码。而后端对其进行进一步加密的原因主要是增强安全性，而加盐的操作最好在后端加密过程中进行。完整的加密流程如下图所示：

****

如流程图所示，前端对密码进行加密后，后端对密文进行了加盐以及二次加密，然后把二次加密的密码和数据库中存储的用户密码进行比对，如果一致则判定用户输入密码成功。

另一个问题是如何选择盐，从安全性的角度考虑，每个用户应该有属于自己的盐，如果所有用户采用统一的盐，在数据库数据泄露后，更容易被暴力破解密码。而如果每个用户有唯一的盐，则黑客需要针对每个用户做单独的破解，大大提高了破解成本。盐的选择可以用生成随机数的方式，或者可以考虑用每个用户唯一标识的用户ID作为盐，以保证每个用户拥有一个属于自己的盐。

* 1. **其他类加密**

一些其他的加密方法，比如知名的RSA加密，并不适合于密码的加密，RSA是一种非对称加密算法，通过公钥和私钥，分别实现对信息的加密和解密的过程。也就是说通过公钥加密的信息，可以通过私钥解密， 这就和哈希算法完全不同，哈希算法是根本不考虑解密的，因为从设计上说也根本不希望加密的信息可以通过某种算法解密。因此从本质上来说，MD5, SHA-1, SHA-2是一种哈希算法（文档和证书签名），而RSA是一种加密/解密算法（安全通信）。而对于密码的加密显然希望通过比对密文，来确定用户是否输入了正确的密码，而不是将密码还原成密码原文。所以RSA加密一般来说并不适合密码的加密。

1. **经验总结，预防措施核对规范的建议等**

本文主要介绍了几种常见的密码哈希加密方式，包括Base64, MD5, SHA-1, SHA-2，并对加密原理做出了简要说明，对在实际工作中密码加密的流程，前端首次加密，后端加盐并进行二次加密的方式进行了分析和梳理。在今后涉及密码登录的场景，可通过采取更安全的加密算法SHA-2, 以及更安全的加密流程，提高系统安全性。

1. **从本文可导出的检查项（checklist）**

无