第六讲 哈希函数

前言

通过前面的学习，我们知道了怎样解决账户和签名数字化问题，即用公钥密码中的公钥作为账户，用私钥对消息生成数字签名。那么你还记得货币数字化中需要解决的另外一大问题是什么吗？

没错，是账本的去中心化问题。我们还分析了。由于在去中心化的网络中没有中心节点，所以具体来说我们需要解决以下两个方面的问题。

1. **无人记账。**记账显然要耗费计算和存储资源，所以没有人会做损己利人的事；
2. **账本不可信。**既然人人可以记账，那么人人就都可以故意记错账或者篡改被人记的账，这样的分布式账本当然不可信。

所以在没有中心节点的情况下，该设计怎样的机制才能既有人记账，又让账本可信?

接下来，我们先解决账本可信问题，之后再解决谁来记账的问题。

账本可信问题

完整性保护问题

我们所说的账本可信问题，就是账本的完整性保护问题。什么是完整性保护问题呢？

简单来说，就是确保保护对象没有被篡改。

比如，你今天晚上写了一个文案后，存档。当你第二天再次打开电脑时，怎么检验在你昨天晚上离开之后文案没有被人修改过呢？

再比如，当你从某个第三方网站下载一个软件时，你该怎样检验这个软件和官方发布的那个软件一模一样，没有被第三方网站篡改植入恶意代码呢？

这都是完整性保护问题。

完整性保护的初级方案

如果不使用密码学，那么你大概只能按下面的方法检验完整性有没有遭到破坏。

- 当你写完文案，在电脑上存档的同时，复制一份副本保存在你的u盘上。把u盘随身携带，确保不被任何人接触。第二天再次打开电脑后，将电脑上的文案和u盘上的副本逐字对比。若全部一致，说明文案没有被篡改；否则，说明文案完整性遭到破坏。
- 当你从第三方网站下载一个软件后，再从官方网站下载该软件，将两个软件逐位对比。若全部一致，说明第三方网站下载的软件没有被篡改；否则，说明软件完整性遭到破坏。

你肯定已经发现这么做的问题了。

- 逐字对比、逐位对比，当文案或软件体积很大时，计算量未免太大了。
- 既然U盘上有备份，既然已经从官方网站下载了软件，我干嘛还要使用电脑上的文案，干嘛还要使用第三方网站上的软件，这不是多此一举吗？

哈希函数

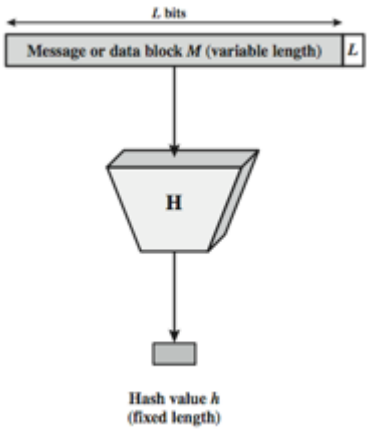
如果使用密码学技术，完整性保护的问题就迎刃而解了。

哈希函数

这个密码学技术就是哈希函数，我们常听说的MD5、SHA256等都是一种哈希函数。

什么是哈希函数呢？

哈希函数是一种以任意长度消息M为输入，以固定长度消息h为输出的一种函数$h = H(M)$，我们常称h为M的哈希值。



密码学哈希函数

不仅如此，而密码学哈希函数必须要具备以下特性。

- **单向性。**由于输入消息可以是任意长度的，所以理论上可能的输入有无穷多种情况；由于输出是固定长度的，所以输入只有有限种情况，所以同一个哈希值h必然对应着多个相同的消息$M_1, M_2, \cdots, M_n$，满足$H(M_i) = h, 1 \leq i \leq n$。所以在理论上哈希函数是不可逆函数。**不仅如此，密码学函数还要求，由已知哈希值h找到$M_1, M_2, \cdots, M_n$中的任意一个，在计算上是不可能的，即单向性。**
- **抗碰撞性。**尽管同一个哈希值h必然对应着多个相同的消息$M_1, M_2, \cdots, M_n$，满足$H(M_i) = h, 1 \leq i \leq n$，**但密码学哈希函数还要求极难找到两个消息具有相同的哈希值，即抗碰撞性。**具体来说，抗碰撞性还分为**抗弱碰撞性**和**抗强碰撞性。**
    - **抗弱碰撞性。**给定消息M，找到一个和它具有相同哈希值的不同消息M'，即$M' \neq M, H(M') = H(M)$，在计算上是不可能的。
    - **抗强碰撞性。**找出任意两个消息M和M'，使其哈希值相同$H(M') = H(M)$，在计算上是不可能的。
- **算法公开。**算法当然要公开，否则需要检验哈希值是否正确的人怎么计算哈希值呢。
- **运算效率高。**运算效率如果很低，计算和检验哈希值是否正确的效率就很低，那哈希函数就没有必要存在了。

由于哈希函数具备以上特性，当我们再遇到消息完整性保护问题时，就可以这样做了。

- 当你写完文案，在电脑上存档的同时，用哈希函数计算文案的哈希值，将哈希值保存在u盘上随身携带，确保不被任何人接触。第二天再次打开电脑后，再次计算电脑上文案的哈希值，并与u盘上的哈希值逐位对比。若全部一致，说明文案没有被篡改；否则，说明文案完整性遭到破坏。
- 软件的官方网站发布下载链接的同时，公布该软件的哈希值。当你从第三方网站下载一个软件后，计算该软件的哈希值，并与官方网站公布的哈希值诸位对比。若全部一致，说明第三方网站下载的软件没有被篡改；否则，说明软件完整性遭到破坏。

如此一来就解决了之前的问题啦。

- 无论文案或软件的体积很大时，他们的哈希值长度是固定的，相对小的多，对比起来效率很高。
- 无需在u盘上备份完整文案，也无需从官方网站额外下载一份软件，只需保留原始文案或软件的哈希值即可。

那么会不会有恶意攻击者篡改了文案或软件，而它们的哈希值不变呢？

放心，由于密码学哈希函数具有单向性和抗碰撞性，所以恶意攻击者要想做到这一点是极难的。

常见密码学哈希函数

MD5

MD5是由著名密码学学家Ronald Linn Rivest于1991年设计的一种哈希函数，全称是Message-Digest Algorithm，即消息摘要算法。Rivest正是大名鼎鼎的RSA公钥密码中的那个R。MD5历经1989年的MD2和1990年的MD4两个版本，它对任意长度消息可以生成16个字节，128位的哈希值。利用MD5算法来进行文件校验的方案被大量应用到软件下载站、论坛数据库、系统文件安全等方面。我国著名密码学家王小云，已经在2004年实现了MD5的弱碰撞，针对某一消息，可以快速构造一个和它MD5值相同的另一条消息，因此MD5作为密码学哈希函数已经不再安全了。

SHA

SHA全称是**Secure Hash Algorithms**，即安全哈希算法，是一种由美国国家标准与技术研究院NIST发布的密码学哈希函数族，是美国的一种信息处理标准FIPS。具体来说包括SHA-0到SHA-3四代。

- **SHA-0**，发布于1993年，哈希值为20个字节，160位。由于存在显著缺陷，所以迅速被SHA-1取代。
- **SHA-1**，发布于1995年，哈希值也是20个字节，160位。由于SHA-1也被发现存在缺陷，所以该标准建议在2010年之后不再使用SHA-1。
- **SHA-2**，发布于2001年，由两个哈希函数组成，即SHA-256和SHA-512，他们分别使用32位字长和64位字长，生成的哈希值正如他们的名字一样，分别是256位和512位。他们还分别有一个删节版，即SHA-224和SHA-384，其哈希值长度分别为224位和384位。
- **SHA-3**，发布于2012年，哈希值长度与SHA-2兼容，但内部结构和之前的SHA家族完全不同。

密码学哈希函数的应用

哈希函数不仅在我们刚才描述的场景中有用，还经常应用于以下场景。

1.消息认证

- 确保消息没有被篡改或重放；
- 确认消息发送方声称的身份。

2.单向口令文件

- 为防止口令泄露，服务器端只存储口令的哈希值，而非口令的明文；

3.入侵检测和病毒检测

- 在入侵检测系统和防病毒系统中存储已知攻击或病毒的哈希值；

哈希函数面临的威胁

哈希函数之所以有这么多用途，主要依赖于其单向性和抗碰撞性。

因此攻击者也主要针对这两个特性实施攻击。

针对单向性的威胁

- **字典攻击。**将哈希函数应用于单向口令文件，可有效防止口令泄露，因为攻击者无法从监听获取窃取到的口令哈希值直接还原出口令。但是，由于人们设置口令时为了方便记忆，并不会完全随机的设置口令，而是优先选择111111、123456、iloveyou等常见口令。因此攻击者可以利用常见口令字典，不断尝试，直至哈希值与口令哈希值一致。
- **彩虹表。**字典攻击本质上还是一种暴力破解，只不过由于有限尝试可能性较大的口令，所以可算作是优化版暴力破解。字典攻击的成功与否取决于字典质量。若字典中不包含真实口令，则不可能破解出来；若真实口令在字典中排序靠后，则要花很长时间才能破解出来。因此，有人便利用线下时间，计算大量消息的哈希值，并记录在一个表中，这个表称为彩虹表。当监听或窃取到某个哈希值时，可以从该表中快速查询该哈希值对应的消息。

针对抗碰撞性的威胁

- **利用哈希算法缺陷碰撞。**这种方法主要是发现并利用哈希函数自身存在的缺陷，进行弱碰撞或强碰撞。例如，王小云对MD5的“破解”就是发现并利用MD5的缺陷进行的一种弱碰撞。
- **生日攻击。**这种方法利用哈希值只有有限多种可能进行碰撞，是对哈希函数的一种通用攻击。如同23个人中有人生日相同的概率超过0.5一样，若哈希值长度为n位，则$2^{\frac{n}{2}}$个不同消息中有两个消息的哈希值相同的概率也超过0.5。因此。这也正是这种攻击命名为生日攻击的原因。

小结

我们通过本堂课学习了密码学哈希函数的特性及其常见应用。

无论哈希函数应用于哪一种场景，其本质上都是将原消息比对问题压缩到了哈希值比对问题，从而降低难度。同时依赖于单向性和抗碰撞性来确保安全。

在区块链中，人人都可以记账，该怎样运用哈希函数来保护账本的完整性呢？下次课继续为您讲解。