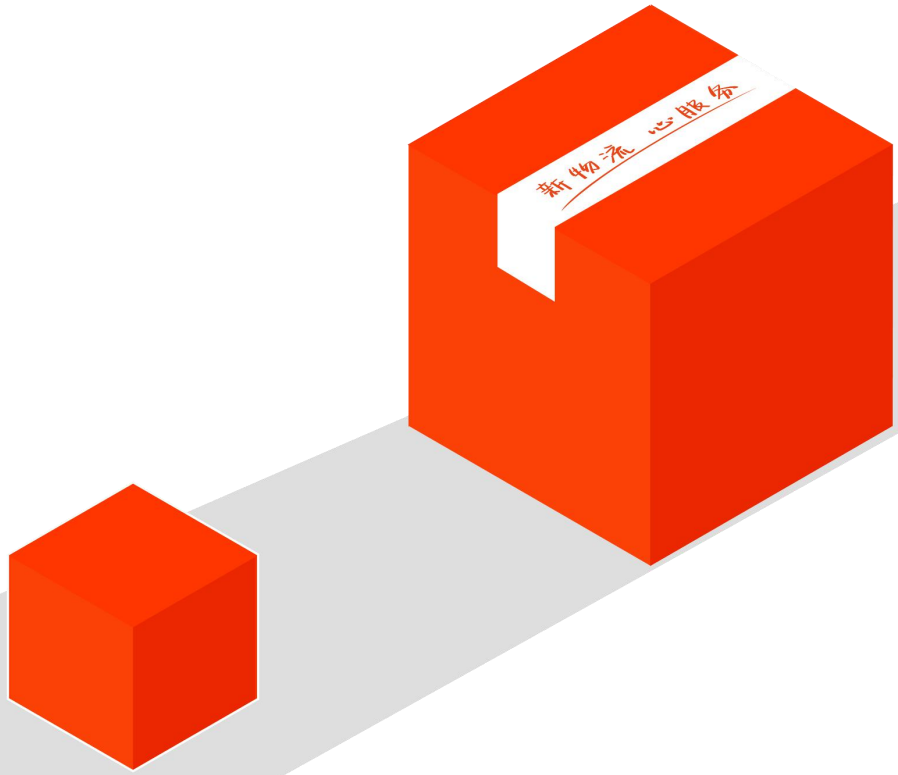
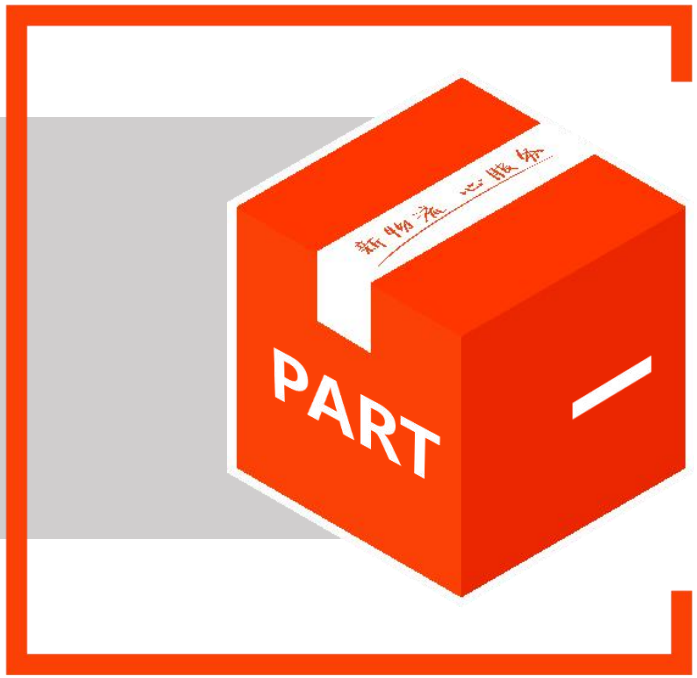
现代密码学

目录

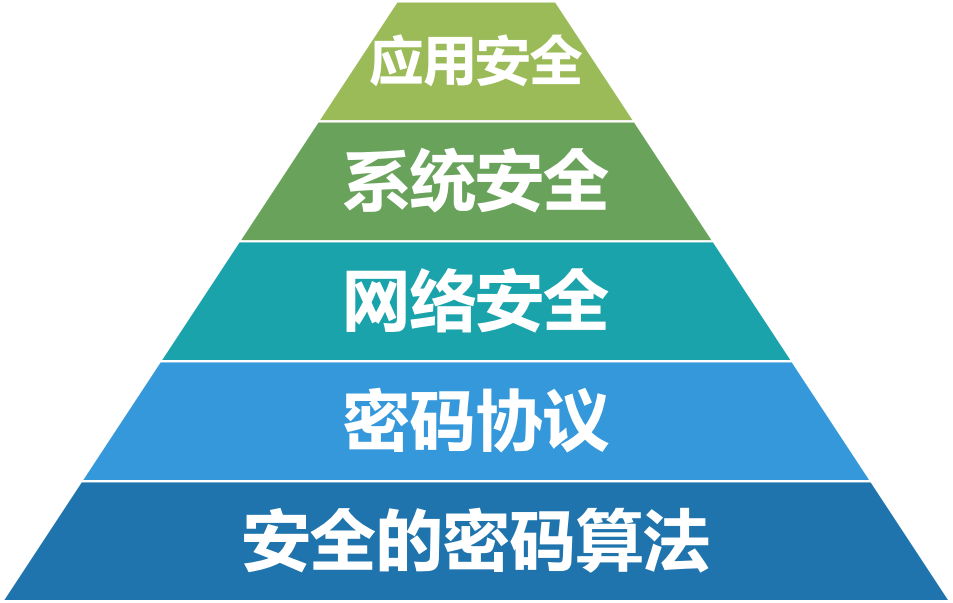




信息安全所面临的威胁

为什么要讲密码学

一、现代密码在社会中的广泛应用



二、密码学在信息安全中的地位

信息安全面临哪些威胁

```
{
""orders"": [{
""upSyncID"": ""88005027018200"",
""saleOrderCode"": ""P6263506759199891 81"",
""createBy"": 0,
""orderDetailList"": [{
""commodityID"": 329301388,
""amount"": 5
        }]
    }]
}
```

OMS=>WMS 接口交互报文

1. 窃听 泄露秘密。
2. 篡改 修改信息。
3. 伪装 伪装成真正发送者
4. 否认 发送者否认。

什么是现代密码学

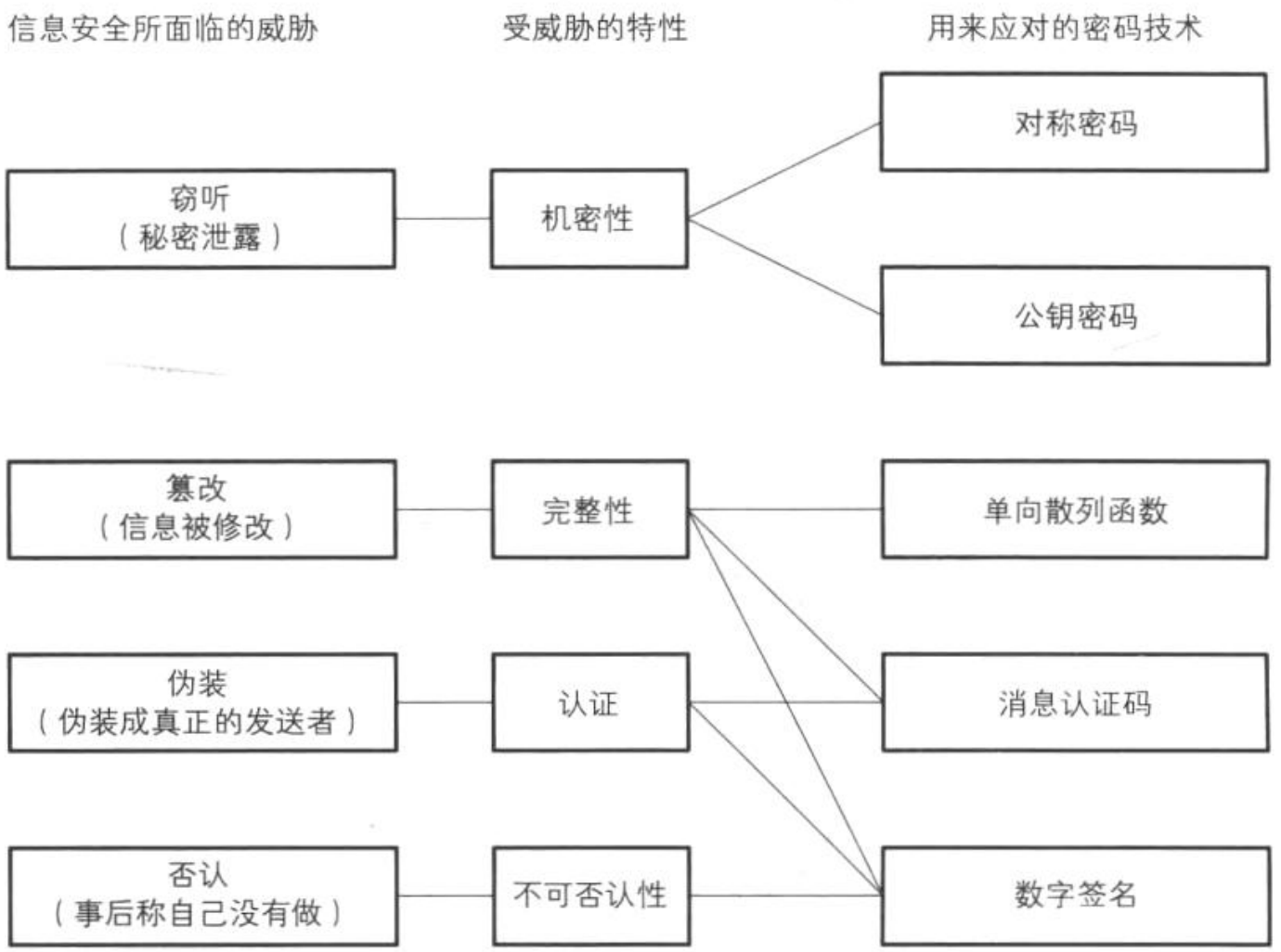
1883年Kerckhoffs第一次明确提出了密码编码的原则：

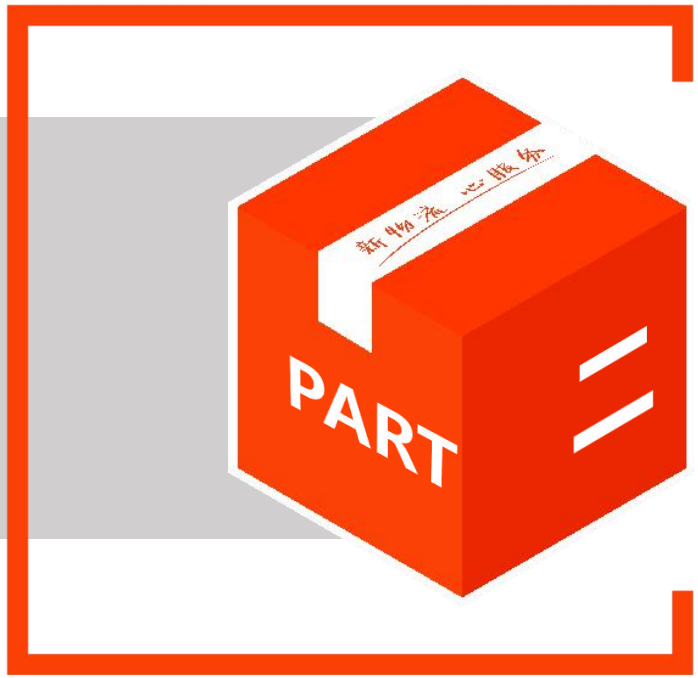
柯克霍夫原则：加密算法应建立在算法的公开不影响明文和密钥的安全，即密码算法的，安全性仅依赖于对密钥的保密。

这一原则已得到普遍承认，成为判定密码强度的衡量标准，也成为古典密码和现代密码的分界线之一。

$$c = E_{k1}(m) \qquad m = D_{k2}(c)$$

如何解决所面临的威胁


信息安全所面临的威胁
受威胁的特性
用来应对的密码技术
窃听
（秘密泄露）
机密性
对称密码
公钥密码
篡改
（信息被修改）
完整性
单向散列函数
伪装
（伪装成真正的发送者）
认证
消息认证码
否认
（事后称自己没有做）
不可否认性
数字签名



对称密码

DES（Data Encryption Standard）

DES（Data Encryption Standard）

1977年得到美国政府的正式许可，是IBM研究出的一种被世界公认，广为流传的一种分组密码算法，公开、细节透明。对推动密码理论的发展起到了重大作用。

1. 分组长度为64 比特。
2. 密钥长56比特。

DES C#代码实现

```
public static string DesEncrypt(string encryptString)
{
    byte[] keyBytes = Encoding.UTF8.GetBytes(key.Substring(0, 8));
    byte[] keyIV = keyBytes;
    byte[] inputByteArray = Encoding.UTF8.GetBytes(encryptString);
    DESCryptoServiceProvider provider = new DESCryptoServiceProvider();
    ICryptoTransform transform = provider.CreateEncryptor(keyBytes, keyIV);
    byte[] data = transform.TransformFinalBlock(inputByteArray, 0, inputByteArray.Length);
    return Convert.ToBase64String(data);
}
```

```
public static string DesDecrypt(string decryptString)
{
    byte[] keyBytes = Encoding.UTF8.GetBytes(key.Substring(0, 8));
    byte[] keyIV = keyBytes;
    byte[] inputByteArray = Convert.FromBase64String(decryptString);
    DESCryptoServiceProvider provider = new DESCryptoServiceProvider();
    ICryptoTransform cTransform = provider.CreateDecryptor(keyBytes, keyIV);
    byte[] data = cTransform.TransformFinalBlock(inputByteArray, 0, inputByteArray.Length);
    return Encoding.UTF8.GetString(data);
}
```

D+mSHZx2nuAEcgxwKtlhtQZAoQiXlYJ2uQCy0sqjIckyDjlAWQkEjj9qTv6D2NuDW3uoxv5cqGuoQXcKl7yj710HbnLmrWip5zynvmSVDPD5h0NLsxUX0bQwVwtzmk3PJdvyYQAIs3I0C4rVbmO17duoEWBwHZN6BxVw+RGQvAmWVyfoRJlprSNaXSuHrJ2MFWnX0QcYdr8zgi6VHZyXb8s8jFXnmVBCF+bGYek+A6sDqNPz0vsCx/iLM6W9DQoT

key：fineex12

AES（Advanced Encryption Standard）

AES（Advanced Encryption Standard）

DES算法由于其密钥较短，难以抵抗现有的攻击，因此不再作为加密标准。

1997年1月，美国NIST向全世界密码学界发出征集21世纪高级加密标准，并成立了AES标准工作研究室。

2000年10月2日，NIST宣布Rijndael作为新的AES。

1. 明文分组可变，128、192、256比特。
2. 密钥长度可变，各自可独立指定为128、192、256比特。

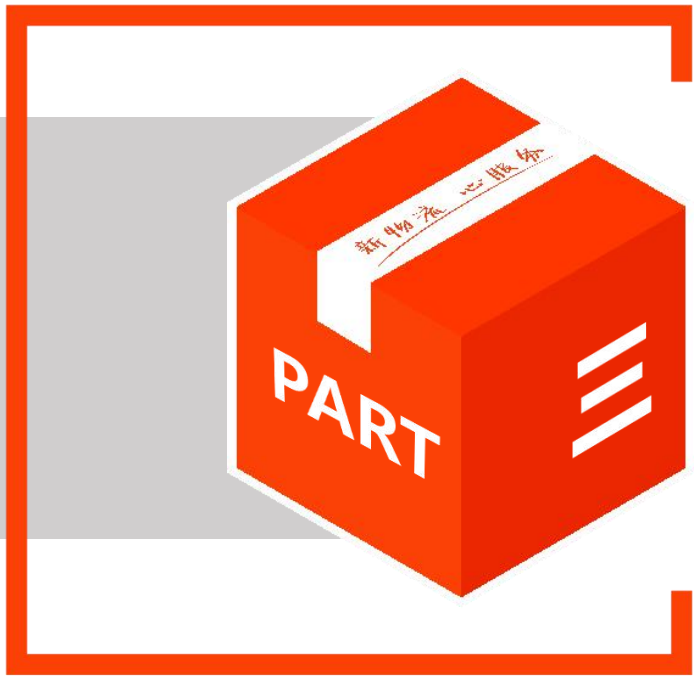
AES C#代码实现

```
public static string AESEncrypt(string encryptStr)
{
    byte[] keyBytes = Encoding.UTF8.GetBytes(Key.Substring(0, 16));
    byte[] keyIV = keyBytes;
    byte[] toEncryptArray = Encoding.UTF8.GetBytes(encryptStr);
    RijndaelManaged rDel = new RijndaelManaged();
    ICryptoTransform cTransform = rDel.CreateEncryptor(keyBytes, keyIV);
    byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0, toEncryptArray.Length);
    return Convert.ToBase64String(resultArray, 0, resultArray.Length);
}
```

```
public static string AESDEncrypt(string encryptStr)
{
    byte[] keyBytes = Encoding.UTF8.GetBytes(Key.Substring(0, 16));
    byte[] keyIV = keyBytes;
    byte[] toEncryptArray = Convert.FromBase64String(encryptStr);
    RijndaelManaged rDel = new RijndaelManaged();
    ICryptoTransform cTransform = rDel.CreateDecryptor(keyBytes, keyIV);
    byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0, toEncryptArray.Length);
    return Encoding.UTF8.GetString(resultArray);
}
```

L52e9m93/e0Lgc/5rEa5pwl0xCoSUih
QlXoOyxmEcGVGf7lJJCXuAHQBbqe
77oTdSLUW40J1wQrFlQbPvA7zq1+
UsRCHWRL6jMe/kAajlbFrMxLAgTbrl
VdW5pla/1bphIsvAuA6zl9bJCtbB+h
8TMoxpHlhNtfuuc8kX6uxMOogIsTQ
8tgtfLRkFBzHyWd0OyJiQobtKES+ow
APDPWBweRPObq5YQK/Dq0jK1fd7
qlowJt8HyVWLi1E7wEfYNRY

key：fineexfineex1234



非对称密码

非对称密码

- 每个用户一个密钥对，一个公钥public key和一个对应的私钥private key。
- 公钥将在系统内被公开。
- 私钥由用户本人安全保管。
- 非对称密码也被称为： 公钥密码。

$$c = E_{k1}(m) \qquad m = D_{k2}(c)$$

RSA

RSA 是1977年由罗纳德·李维斯特（Ron Rivest）、阿迪·萨莫尔（Adi Shamir）和伦纳德·阿德曼（Leonard Adleman）一起提出。

RSA允许选择公钥的大小，目前被破解的最长RSA密钥是768个二进制位，1024位的RSA密钥基本安全，2048位的密钥极其安全。

是迄今未知理论上最为成熟完善的公钥密码体制。

$$c = m^E \ mod \ N \qquad\qquad m = c^D \ mod \ N$$

公钥：E,N

私钥：D,N

RSA算法原理

RSA C#代码实现

```
public string RSAEncrypt(string xmlPublicKey, string content)
{
    string encryptedContent = string.Empty;
    using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider())
    {
        rsa.FromXmlString(xmlPublicKey);
        byte[] encryptedData = rsa.Encrypt(Encoding.Default.GetBytes(content), false);
        encryptedContent = Convert.ToBase64String(encryptedData);
    }
    return encryptedContent;
}
```

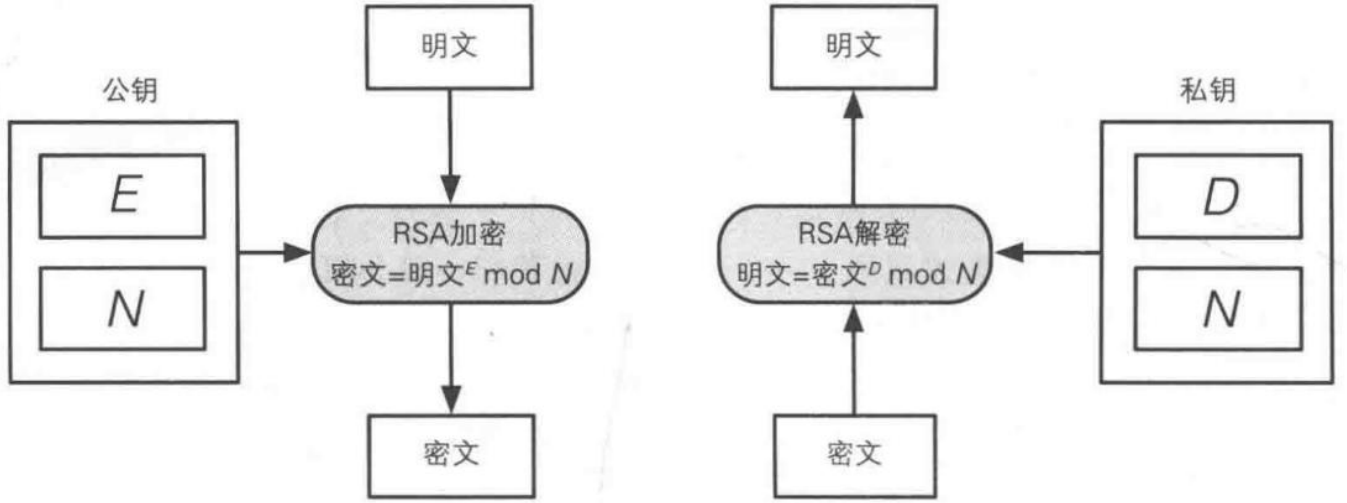
```
public string RSADecrypt(string xmlPrivateKey, string content)
{
    string decryptedContent = string.Empty;
    using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider())
    {
        rsa.FromXmlString(xmlPrivateKey);
        byte[] decryptedData = rsa.Decrypt(Convert.FromBase64String(content), false);
        decryptedContent = Encoding.GetEncoding("gb2312").GetString(decryptedData);
    }
    return decryptedContent;
}
```

明文： fineex

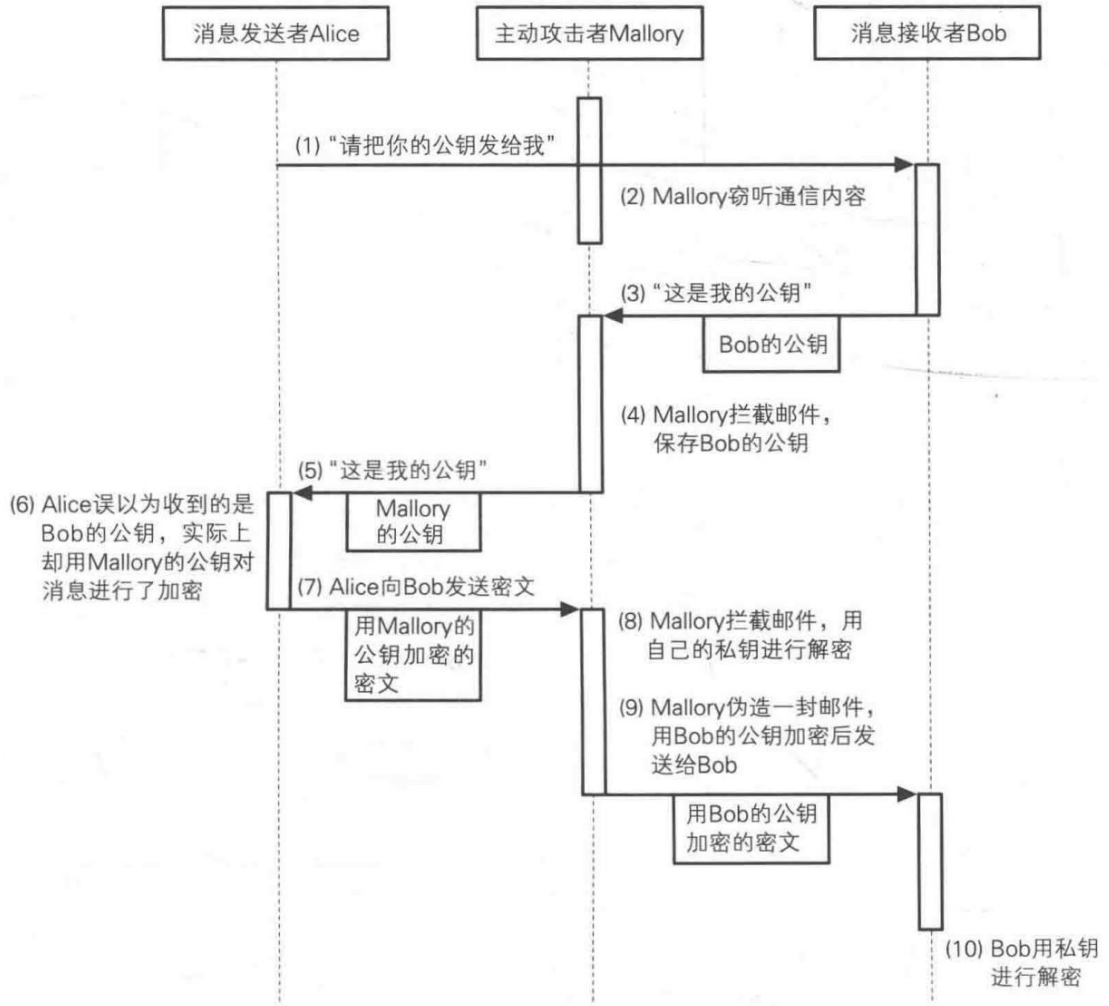
FikVVHgRCL5jhxO6Jner
r3bV4KiHhjnVosy6qLuF
avuJXk48Zukp64cXWN
SGM+X85U5PV0f8eDb
q4A28fi6ifMokERtjgVqy
PzbtGzGSxuMklwIEn9P
+1yg8kRfzv3EPW8cgs4
KRBpfHoHLp5CBckD7b
OOQDbzAV9ggJTqFhlc
M=

非对称密码解决什么问题

非对称密码解决了密钥配送的问题？



中间人攻击(MITM)



中间人攻击（Man-in-the-middle attack）指攻击者与通讯的两端分别创建独立的联系，并交换其所收到的数据，使通讯的两端认为他们正在通过一个私密的连接与对方直接对话，但事实上整个会话都被攻击者完全控制。

单向散列函数

单向散列函数（Hash Function）

单向散列函数也称为消息摘要函数、哈希函数或者杂凑函数，输出的散列值称为消息摘要或者指纹。

一、任意长度消息计算出固定长度的散列值。

二、能够快速计算。

三、消息不同，散列不同。

四、具备单向性，无法通过散列值反算出原消息。

常见单向散列函数(Hash函数):
MD5（Message Digest Algorithm 5）：可以用来把不同长度的数据块进行暗码运算成一个128位的数值。
SHA（Secure Hash Algorithm）这是一种较新的散列算法，可以对任意长度的数据运算生成一个160位的数值。

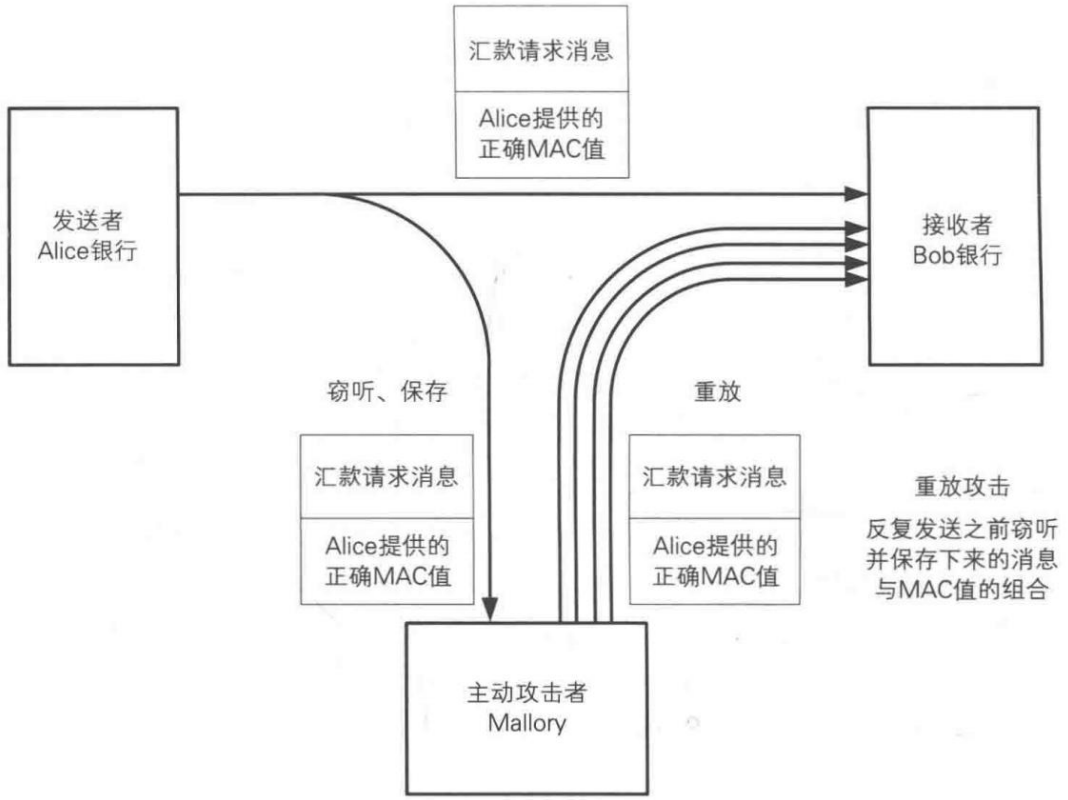
消息认证码（MAC）

消息认证码的输入包括任意长度的消息和一个发送者和接收者之间共享的密钥，可以输出固定长度的数据，这个数据成为MAC值。

$$MAC = C_k(M)$$

通过计算MAC与传值MAC对比，保证了消息不被篡改，保证了发方不是冒充的。

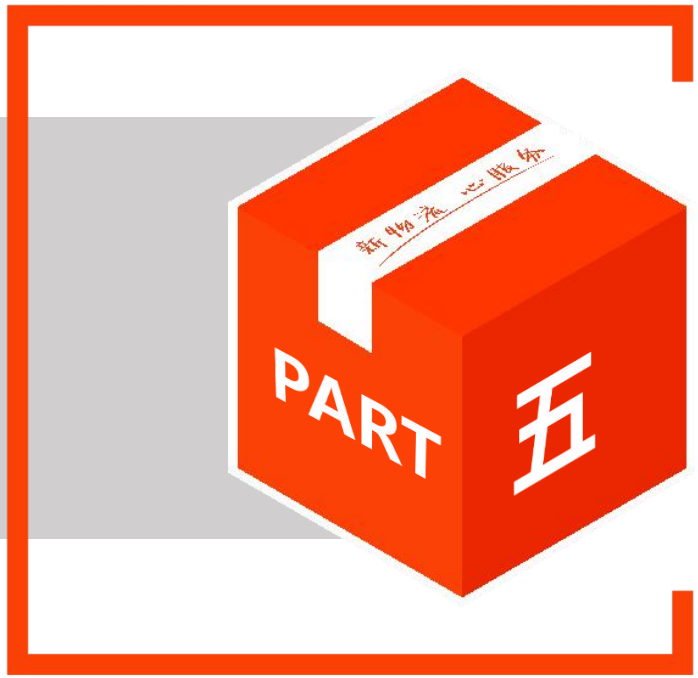
重放攻击(Replay Attack)



重放攻击（Replay Attack）攻击者保存消息和MAC，重复发送。

防范方法：

1. 序号
   每次发送一个递增编号。
2. 时间戳
   消息包含时间戳，可以一定程度上防御。
3. nonce
   事先发送一个随机数。



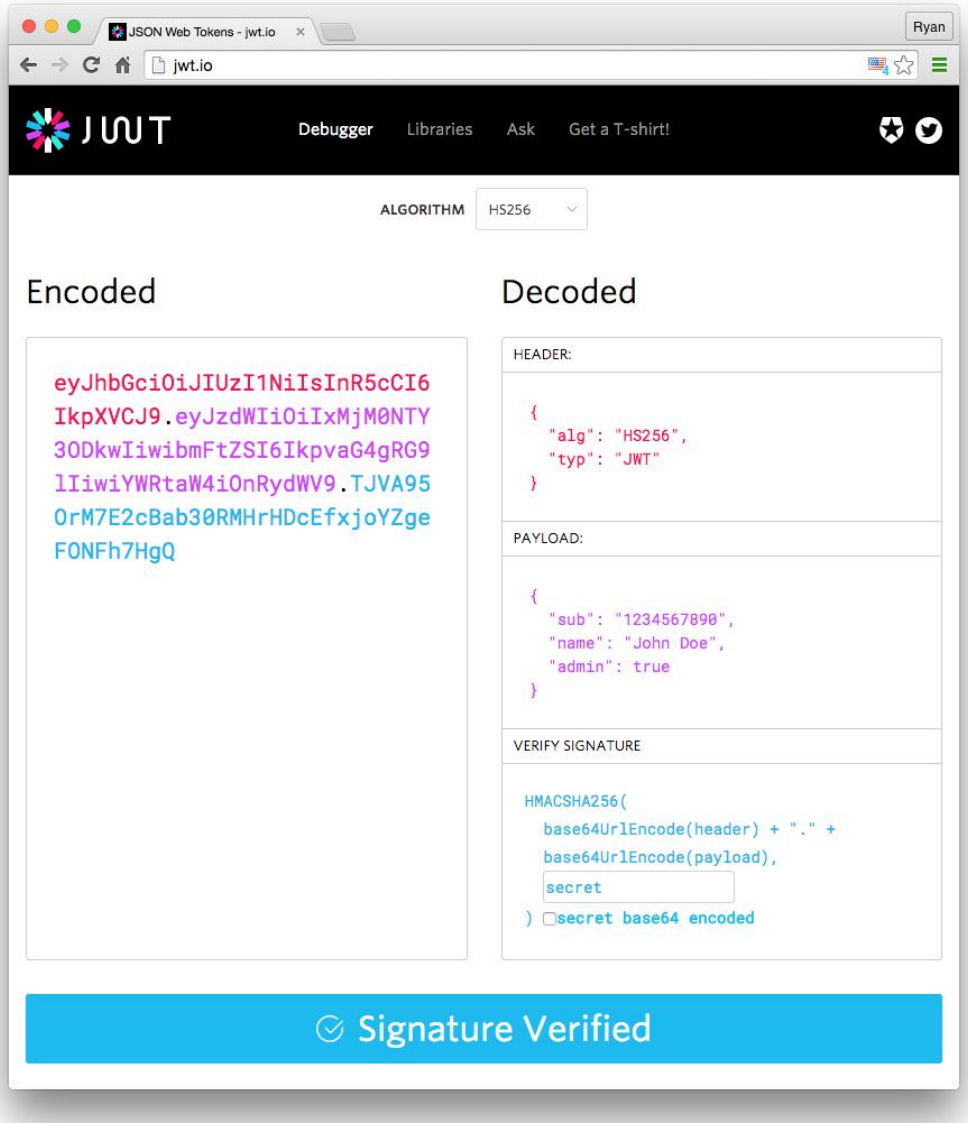
实际应用场景

接口通信

```
public string SignTopRequest(IDictionary<string, string> parameters, string body, string secret)
{
    // 第一步：把字典按Key的字母顺序排序
    IDictionary<string, string> sortedParams = new SortedDictionary<string, string>(parameters, StringComparer.Ordinal);
    // 第二步：把所有参数名和参数值串在一起
    StringBuilder query = new StringBuilder();
    query.Append(secret);
    foreach (KeyValuePair<string, string> kv in sortedParams)...
    // 第三步：把请求主体拼接在参数后面
    if (!string.IsNullOrEmpty(body))
    {
        query.Append(body);
    }
    // 第四步：使用MD5加密
    query.Append(secret);
    MD5 md5 = MD5.Create();
    var bytes = md5.ComputeHash(Encoding.UTF8.GetBytes(query.ToString()));
    // 第五步：把二进制转化为大写的十六进制
    StringBuilder result = new StringBuilder();
    for (int i = 0; i < bytes.Length; i++)
    {
        result.Append(bytes[i].ToString("X2"));
    }
    return result.ToString();
}
```

中台（WMS）接口通信加密方法

JWT



JWT (JSON Web Token)

SIGNATURE = HMACSHA256(

base64UrlEncode(header) + "." +

base64UrlEncode(payload),

secret

)

[jwt.io](jwt.io)

证书(PKC)

1. 怎么获得对方的公钥？

公钥证书（Public Key Certificate），里面记有个体及个体公钥，并且有认证机构（CA）施加了数字签名，简称证书。

2. 怎么获得认证机构的公钥?

对于用来验证数字签名的认证机构的公钥，需要由其他认证机构施加签名的公钥证书。

证书链，终点 根CA(Root CA)。

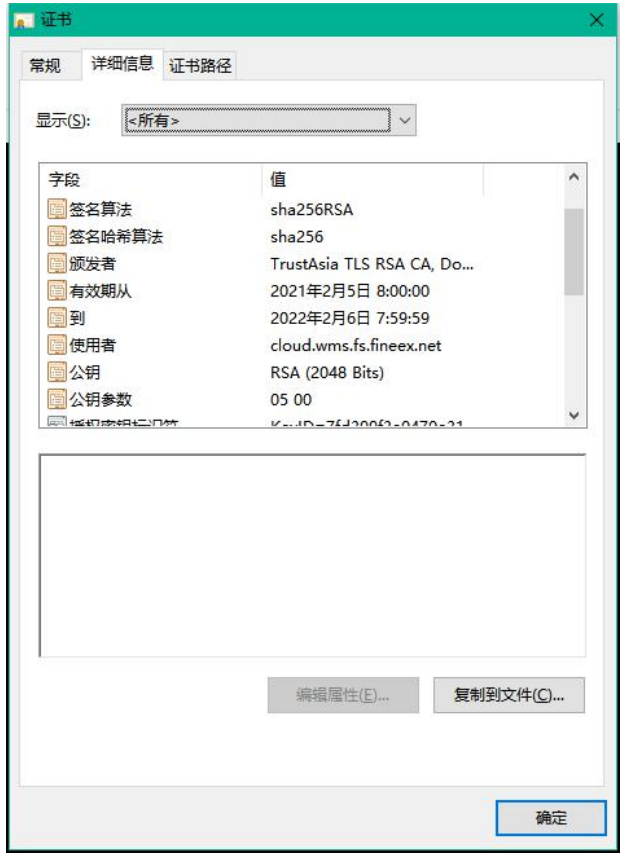
超文本传输安全协议（HTTP over TLS/ SSL）

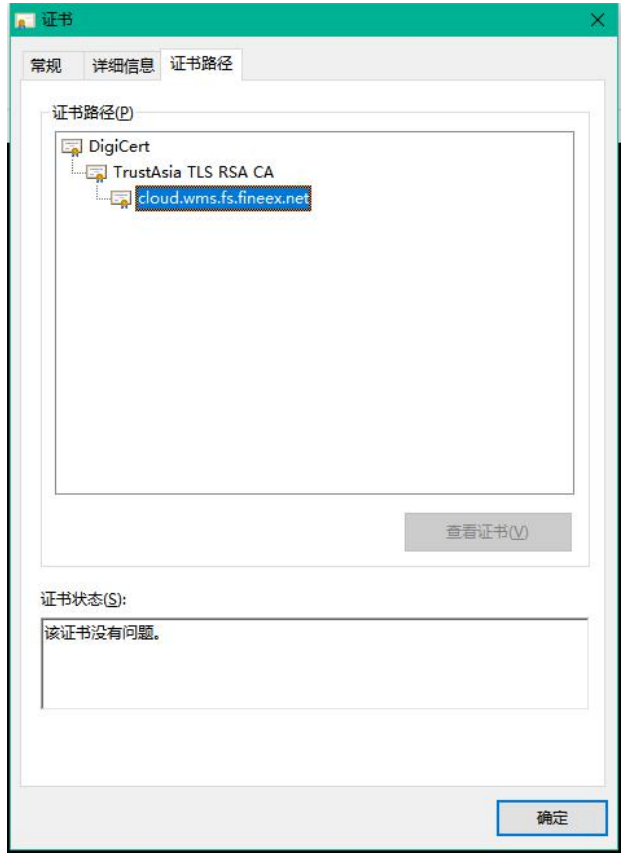
超文本传输安全协议（HTTP over TLS/ SSL）

一种通过计算机网络进行安全通信的传输协议，经由HTTP进行通信，但利用SSL/TLS来加密数据包，简称HTTPS。

HTTPS开发的主要目的，是提供对网站服务器的身份认证，保护交换资料的隐私与完整性。

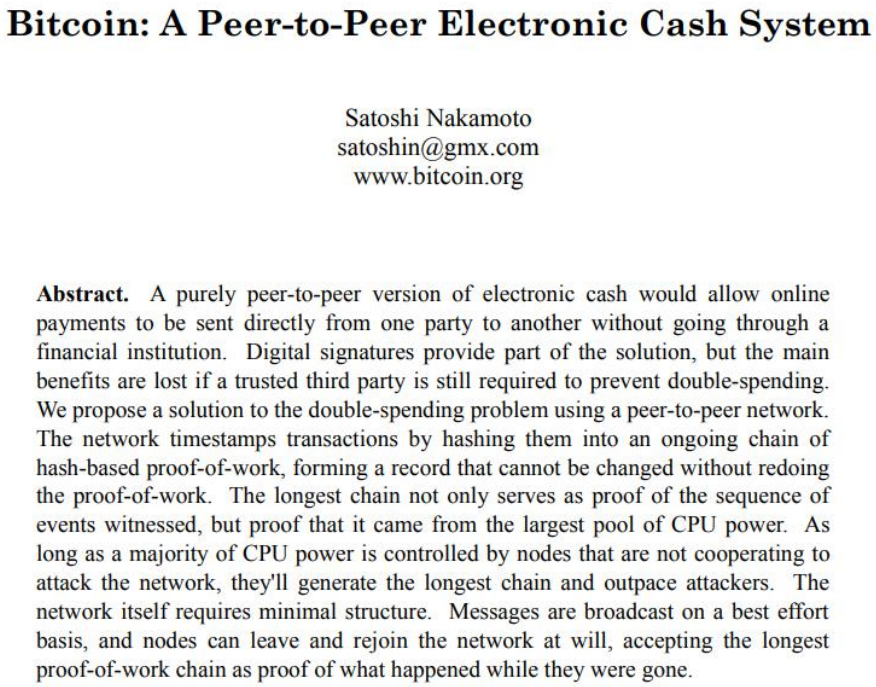
数字证书（Digital Certificate）





证书明细和证书链

比特币（BitCoin）



Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

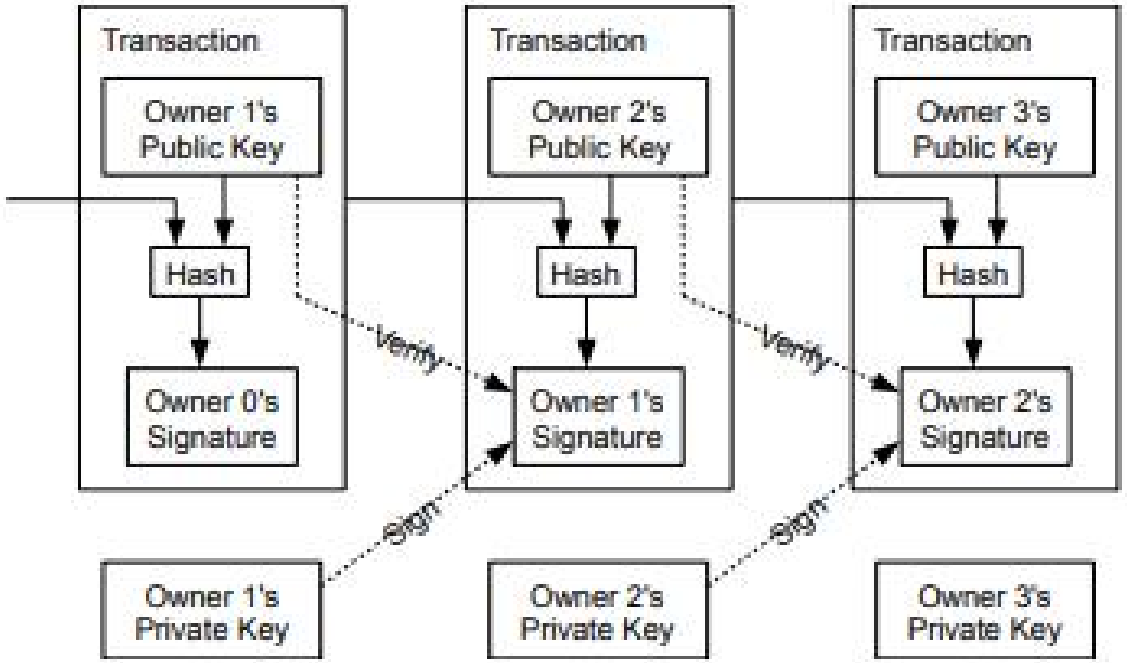
Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

比特币（Bitcoin）

最初由中本聪在2008年提出，并于2009年1月3日正式诞生 ,是一种P2P形式的虚拟的加密数字货币。点对点的去中心化的支付系统。

2015年,美国开设了世界上第一家比特币交易所Coinbase。

交易（Transactions）

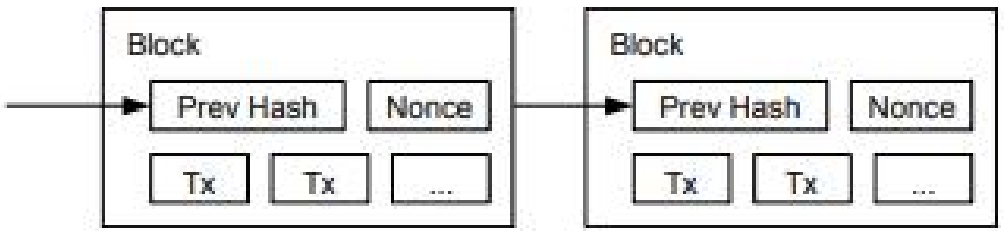


交易（Transactions）

一枚电子货币就是一条数字签名链。每个拥有者都通过将上一次交易和下一个拥有者的公钥的哈希值的数字签名添加到此货币末尾的方式将这枚货币转移给下一个拥有者。

收款人可以通过验证数字签名来证实其为该链的所有者。

工作量证明（Proof-of-Work)



区块链（Block Chain）

工作量证明（Proof-of-Work）

工作量证明采取搜索一个数（Nonce），使得被哈希时，如使用 SHA-256，得到的哈希值以数个 0 比特开始。

改变这个区块将需要重做所有后面的区块。

谢谢观看