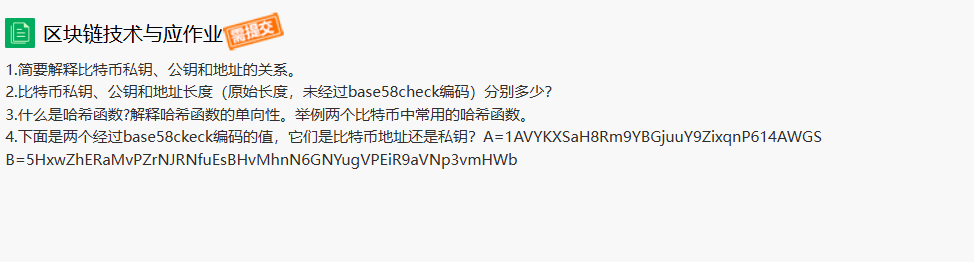
## 第一次作业



https://www.hxtop.com/Plug/online-btc-address.html  
比特币地址在线生成

1. \*\*比特币私钥、公钥和地址的关系\*\*：

比特币私钥是随机生成的 256 位数字，通过非对称加密算法生成对应的公钥。公钥是 512 位的数字，它和私钥是一对，通过私钥可以计算出公钥，但反过来几乎不可能。

\*公钥经过一系列的哈希运算和编码后可以得到比特币地址。地址是一个较短的、易于使用的字符串，用于接收比特币。简单来说，私钥是用于签名交易；公钥用于验证签名；地址是公钥经过处理后用于接收比特币的标识。

2. \*\*比特币私钥、公钥和地址长度（原始长度，未经过 base58check 编码）\*\*：

\* 比特币私钥的原始长度是 256 位（32 字节）。

\* 比特币公钥的原始长度有两种情况，非压缩公钥长度是 64 字节（512 位），压缩公钥长度是 33 字节。

\* 比特币地址在未经过 base58check 编码前，是其原始哈希形式（，长度是 20 字节（160 位）。

3. \*\*哈希函数及其单向性\*\*：

\* 哈希函数是一种数学函数，它接受任意长度的输入数据，通过算法计算后，输出一个固定长度的字符串。常用的哈希函数有 SHA - 256 和 RIPEMD - 160。

\* 哈希函数的单向性是指对于给定的输入，很容易通过哈希函数计算出输出哈希值，但是从输出的哈希值几乎不可能反推出原始的输入数据。

4. \*\*判断 A 和 B 是比特币地址还是私钥\*\*：

\* 比特币私钥通常经过 base58check 编码后以 “5” 开头（未压缩的 WIF 格式），或者以 “K” 或 “L” 开头（压缩的 WIF 格式）。

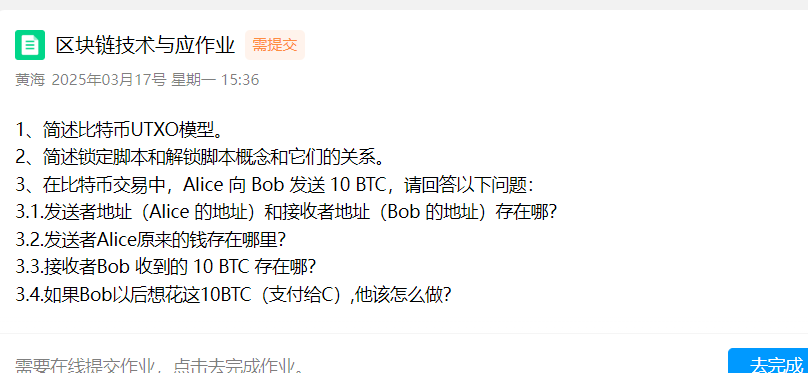
比特币地址通常以 “1” 开头（P2PKH 地址）或者 “3” 开头（P2SH 地址）。

\* 根据给出的值，A = 1AVYKXSaH8Rm9YBGjuuY9ZiqnP614AWGS 以 “1” 开头，符合比特币 P2PKH 地址的格式，所以 A 是比特币地址。

\* B = 5HxwZheERaMvPZrNJRNFuEsBHVmMhnN6GNYugVPEiR9aVNp3vmHWb 以 “5” 开头，符合未压缩的 WIF 格式的比特币私钥格式，所以 B 是比特币私钥。

复制公钥

就是公钥＋链码



以下是针对作业问题的详细解答：

### \*\*1. 简述比特币UTXO模型\*\*

- \*\*UTXO（Unspent Transaction Output）\*\*：比特币使用UTXO模型来记录交易。UTXO代表未花费的交易输出，是比特币交易的基本单位。

- \*\*工作原理\*\*：

- 每笔交易的输入是之前交易的UTXO，输出是新的UTXO。

- 交易完成后，输入UTXO被标记为已花费，输出UTXO成为新的未花费输出。

- \*\*特点\*\*：

- 比特币网络通过UTXO集合跟踪所有未花费的输出。

- UTXO模型支持并行交易处理，提高交易效率。

### \*\*2. 简述锁定脚本和解锁脚本概念和它们的关系\*\*

- \*\*锁定脚本（ScriptPubKey）\*\*：

- 定义UTXO的解锁条件，通常包含接收者的公钥哈希。

- 示例：`OP\_DUP OP\_HASH160 <Bob的公钥哈希> OP\_EQUALVERIFY OP\_CHECKSIG`。

- \*\*解锁脚本（ScriptSig）\*\*：

- 提供满足锁定脚本条件的数据，通常包含签名和公钥。

- 示例：`<Alice的签名> <Alice的公钥>`。

- \*\*关系\*\*：

- 解锁脚本与锁定脚本组合后执行，验证交易合法性。

- 若脚本执行成功，UTXO被解锁，交易有效。

### \*\*3. 比特币交易问题解答\*\*

#### \*\*3.1 发送者地址（Alice 的地址）和接收者地址（Bob 的地址）存在哪？\*\*

- \*\*发送者地址\*\*：存在于交易输入的锁定脚本中（即Alice之前收到的UTXO的锁定脚本）。

- \*\*接收者地址\*\*：存在于交易输出的锁定脚本中（即Bob的新UTXO的锁定脚本）。

#### \*\*3.2 发送者Alice原来的钱存在哪里？\*\*

- Alice的钱存在于她之前收到的UTXO中。这些UTXO记录在区块链上，作为她钱包的余额。

#### \*\*3.3 接收者Bob 收到的 10 BTC 存在哪？\*\*

- Bob收到的10 BTC存在于新的UTXO中，该UTXO记录在区块链上，锁定脚本包含Bob的公钥哈希。

#### \*\*3.4 如果Bob以后想花这10BTC（支付给C），他该怎么做？\*\*

- \*\*步骤\*\*：

1. \*\*创建交易\*\*：Bob使用他的私钥生成解锁脚本，引用他收到的10 BTC UTXO作为输入。

2. \*\*指定输出\*\*：创建新的UTXO，锁定脚本包含C的公钥哈希。

3. \*\*广播交易\*\*：将交易广播到比特币网络，等待矿工确认。

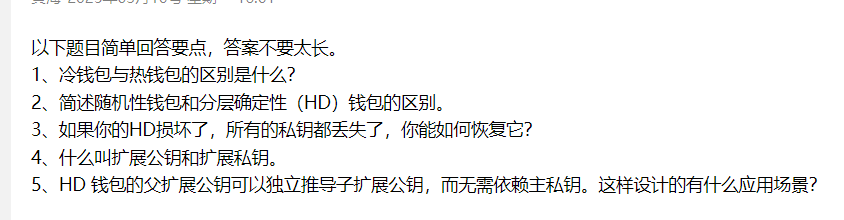
4. \*\*交易确认\*\*：交易被打包进区块后，C收到10 BTC。

### \*\*总结\*\*

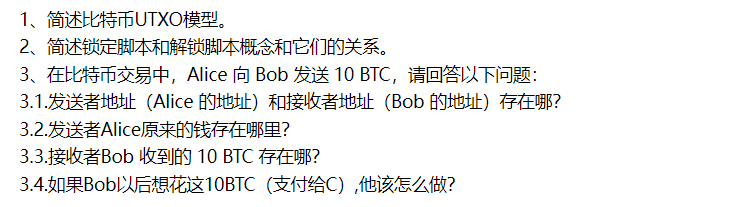
- UTXO模型是比特币交易的核心机制，通过锁定脚本和解锁脚本实现交易验证。

- 交易中，地址和资金信息存储在UTXO的锁定脚本中，接收者通过解锁脚本使用资金。

## 第二次作业



## 第三次作业



1、UTXO模型是记录未花费比特币交易的方式，追踪未花费的比特币输出，确保不发生双重花费

2、锁定脚本：定义UTXO花费的条件

解锁脚本：满足锁定脚本的条件

只有解锁脚本和锁定脚本匹配，UTXO才能花费

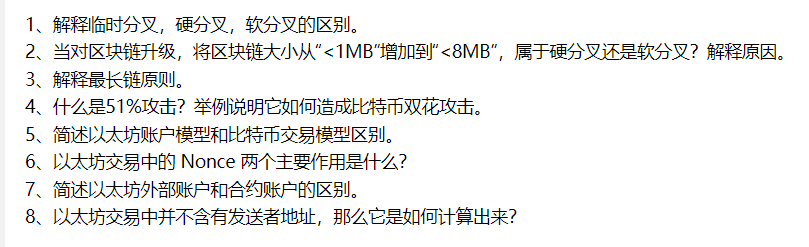
3.1、发送者地址：存在交易输入部分，指向Alice收到的UTXO

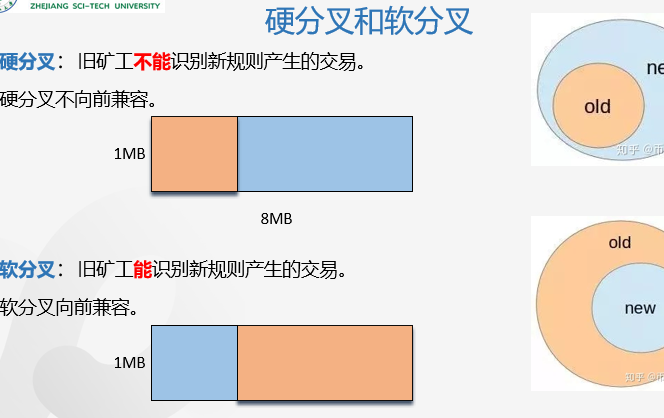
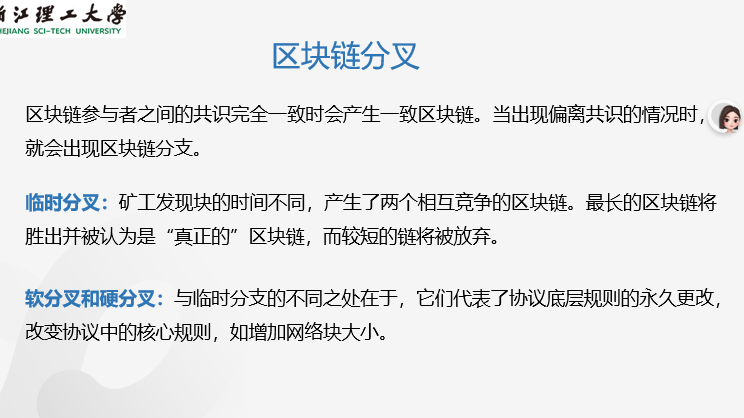
接受者地址：存在交易输出部分，

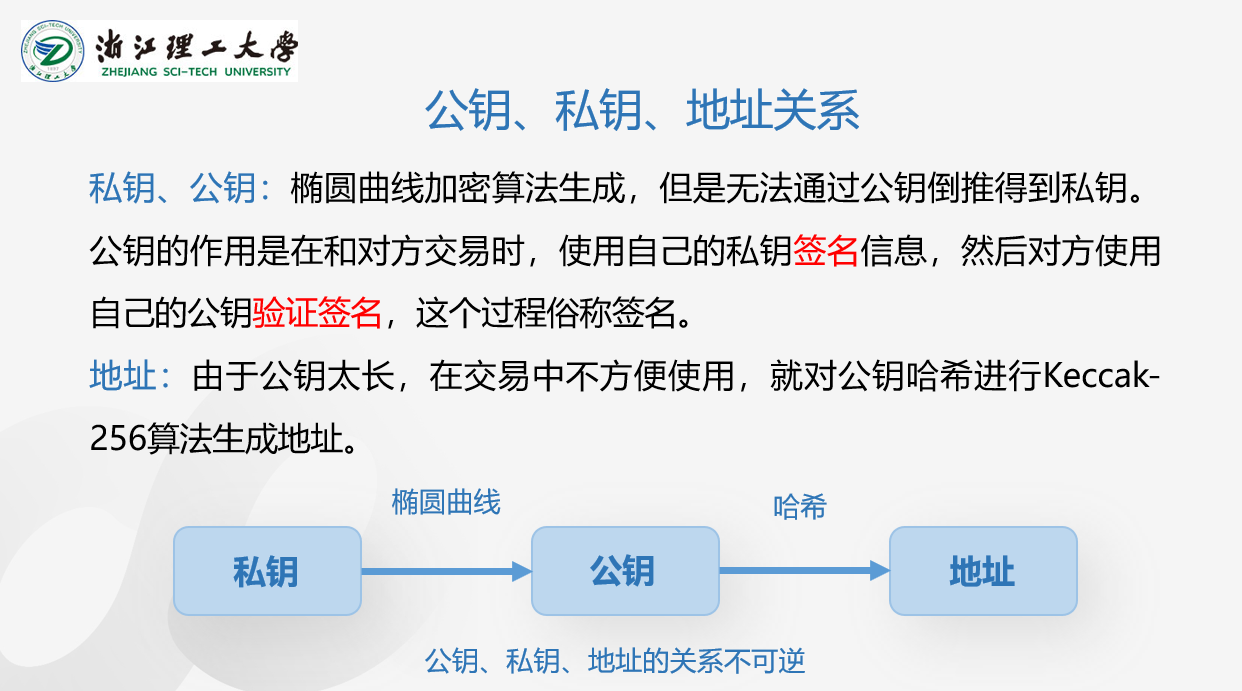
3.2、钱存在UTXO中，UTXO作为这次交易的输入。

3.3、存在新的UTXO中

3.4、Bob需要创建新的交易，用UTXO作为交易输入，C的地址作为交易输出，用解锁脚本（他的签名和公钥）确保符合UTXO锁定脚本。

第五次作业：  






1. \*\*临时分叉、硬分叉、软分叉的区别\*\*

- \*\*临时分叉\*\*：区块链网络中因节点同步延迟或同时产生多个合法区块，导致区块链暂时出现多个分支。通常通过最长链原则，较短分支会被丢弃，网络最终达成一致。

- \*\*硬分叉\*\*：区块链协议规则发生不向后兼容的改变，新规则节点无法接受旧规则区块，旧节点也无法接受新规则区块，导致区块链永久分裂成两条链（如比特币和比特币现金的分裂）。

- \*\*软分叉\*\*：区块链协议规则发生向后兼容的改变，新规则节点可以接受旧规则区块，但旧节点可能无法完全理解新规则区块。旧节点仍可参与网络，但功能受限（如比特币的隔离见证升级）。

2. \*\*区块链大小从“<1MB”增加到“<8MB”属于硬分叉还是软分叉？\*\*

- \*\*属于硬分叉\*\*。

- \*\*原因\*\*：将区块大小限制从“<1MB”增加到“<8MB”意味着新规则允许包含更多交易数据的区块，而旧规则节点无法接受超过1MB的区块。这种不向后兼容的改变会导致旧节点拒绝新区块，造成网络分裂，因此是硬分叉。

3. \*\*最长链原则\*\*

- 最长链原则是区块链共识机制的核心规则，指节点始终选择累计工作量（或区块数量）最多的链作为主链。当网络出现分叉时，节点会优先接受最长（或最难生成）的链，较短的分支会被丢弃。这确保网络最终达成一致，防止双花等攻击。

4. \*\*51%攻击及比特币双花攻击示例\*\*

- \*\*51%攻击\*\*：攻击者控制超过50%的网络算力，从而能够操控区块链的区块生成，可能篡改交易记录或阻止其他交易确认。

- \*\*双花攻击示例\*\*：攻击者向商家发送一笔比特币支付（交易A），并在区块中确认，商家交付商品。随后，攻击者利用51%算力生成一条更长的链，包含另一笔将相同比特币花到自己地址的交易（交易B），并排除交易A。由于最长链原则，网络接受新链，交易A被逆转，攻击者成功“双花”，既保留商品又拿回比特币。

5. \*\*以太坊账户模型和比特币交易模型的区别\*\*

- \*\*比特币交易模型\*\*：基于UTXO（未花费交易输出）。每笔交易引用之前的UTXO作为输入，生成新的UTXO作为输出。余额通过累加所有未花费的UTXO计算，交易链式追踪资金来源。

- \*\*以太坊账户模型\*\*：基于账户余额。每个账户（外部账户或合约账户）有余额、Nonce等状态，交易直接更新发送者和接收者的账户状态（如余额扣减或增加）。相比UTXO，账户模型更像传统银行账户，简化了状态管理但增加了复杂性（如智能合约）。

6. \*\*以太坊交易中Nonce的两个主要作用\*\*

- \*\*防止重放攻击\*\*：Nonce是账户的交易计数器，每笔交易的Nonce必须递增且唯一，确保同一交易无法被重复执行。

- \*\*保证交易顺序\*\*：Nonce决定了交易的执行顺序，节点按Nonce从小到大处理交易，防止乱序执行或跳过交易。

7. \*\*以太坊外部账户和合约账户的区别\*\*

- \*\*外部账户（EOA）\*\*：由私钥控制，代表用户或实体，只能发起交易（如转账或调用合约），不包含代码，状态仅包括余额和Nonce。

- \*\*合约账户\*\*：由智能合约代码控制，无私钥，包含代码和存储数据，可自动执行逻辑（如条件转账）。交易可触发其代码运行，状态包括余额、Nonce、代码和存储。

8. \*\*以太坊交易中如何计算发送者地址\*\*

- 以太坊交易不直接包含发送者地址，而是包含发送者的签名（v, r, s）。通过以下步骤计算发送者地址：

1. 从交易数据中提取签名（r, s）和恢复参数（v）。

2. 使用椭圆曲线数字签名算法（ECDSA），根据签名和交易内容的哈希恢复出发送者的公钥。

3. 对公钥进行Keccak-256哈希，取最后20字节作为发送者地址。

- 这种机制确保只有拥有对应私钥的用户才能生成有效签名，从而验证发送者身份。