上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

第七次作业

Log Creative

2023 年 12 月 4 日

服务提供商规定一些面值的比特币交易，比如：$100, 50, 10, 5, 1, 0.5, 0.1$，由于服务提供商会知道公钥 PK_i 与用户 i 的对应关系，根据用户需要洗币的金额 v，按照找零算法（先换大面值，再换小面值）计算需要的接收公钥数 n_i，服务提供商会要求用户提供对应数目的接收公钥 $PK'_{ij}(j = 1, \cdots, n)$。服务提供商将会给用户提供服务的接收公钥地址 PK_S。

用户与服务器交换完公钥地址信息后，用户以公钥 PK_i 向服务器地址 PK_S 发送 v 比特币，服务器在接收到其他用户足够的金额的比特币后，会用其他用户发送给服务器产生的 UTXO 发送给用户提供的接收公钥地址 PK'_{ij}，这些交易之间会有相对随机的间隔时间，这样就达到了洗币的目的。服务提供者可能会收取服务费。

为了保障隐私，服务者应该保证公钥与用户的对应信息不被泄漏，即 PK_i 与 PK'_{ij} 的对应关系。并且服务者需要保证不会克扣除服务费外的其他比特币。

为了简化讨论的情形，假设服务提供者不收取服务费。这里还假设可以有除了 A，B，C 外的其他用户，比如 D，E，F 等。因为：

- 服务提供者需要足够多的用户才可以为用户提供足够好的隐私性，以更好地混合，提供更有隐私的服务；
- 使用找零算法可能会产生无法完全匹配的情况，但是好处就是这个算法是确定性的，在用户提交需求时就可以按照预先设定的面额得知需要多少个接收公钥地址，而不是取决于服务器的运营情况；
- 不需要中间服务者的情况可以考虑 Coin-Join，一般 Coin-Join 都会基本让交换的币值相等。如果服务器决定用相同的面值，那么该面值会不可避免地足够小，从而产生过多的交易，而大幅降低服务效率。

回到本题的具体情景，根据找零算法，分拆用户提供的金额：

$$A : 115 = 100 + 10 + 5$$

$$B : 120 = 100 + 10 + 10$$

$$C : 12 = 10 + 1 + 1$$

这样，A、B、C 都应该各自分别提供 3 个接收公钥地址 $PK'_{Aj}(j = 1, 2, 3)$、$PK'_{Bj}(j = 1, 2, 3)$、$PK'_{Cj}(j = 1, 2, 3)$。假设服务器又接收到了 D 的 5 比特币，E 的 1 比特币，F 的 1 比特币，他们也都提供了对应的接收公钥地址 $PK'_{D1}, PK'_{E1}, PK'_{F1}$。

交易最终流向如下：

$$
\begin{aligned}
A &: 115 = 100(PK'_{B1}) + 10(PK'_{B2}) + 5(PK'_{D1}) \\
B &: 120 = 100(PK'_{A1}) + 10(PK'_{A2}) + 10(PK'_{C1}) \\
C &: 12 = 10(PK'_{B3}) + 1(PK'_{E1}) + 1(PK'_{F1}) \\
D &: 5 = 5(PK'_{A3}) \\
E &: 1 = 1(PK'_{C2}) \\
F &: 1 = 1(PK'_{C3})
\end{aligned}
$$

具体的交易图如图 1 所示，其中 (PK_i, v) 表示一“枚”比特币，即公钥地址 PK_i 上的 v 金额的 UTXO。交易 tx 按时间顺序编号，实际上只要保证交易依赖关系树后级的交易在得知地址信息的情况下晚于前级的交易即可。

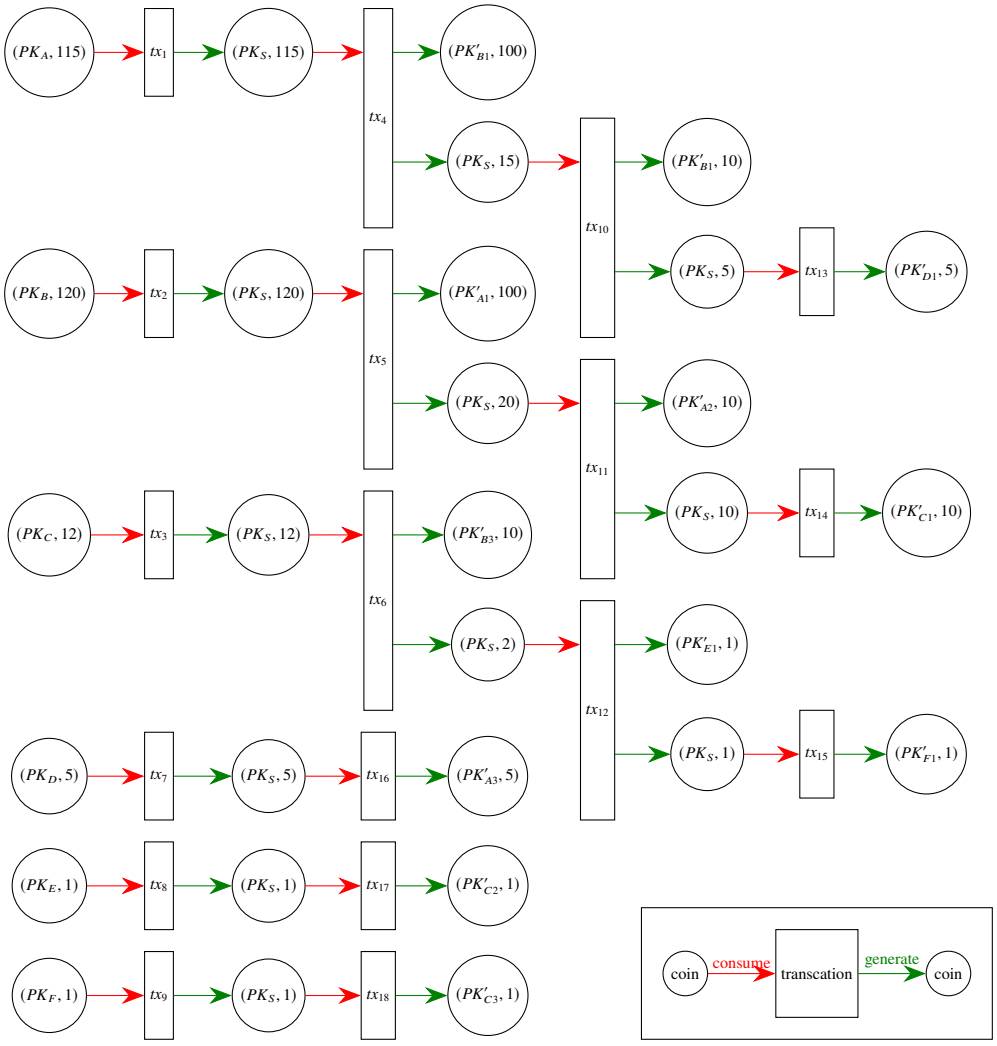

图 1　交易图