问题二

当私密密钥为“01147afe0f4b6332feb1c45aad835c7f89fd272de42974acda47e1f145ac2d89”时，请生成公开密钥。

问题三

用问题二中生成的公开密钥生成地址。

第9章 钱包

在学习区块链时，无法回避钱包的概念。下面我们依次讲解钱包的概念及其演变。

9.1 什么是钱包

钱包并不是存储密码资产本身，而是存储与该资产关联的私密密钥。

9.1.1 钱包管理私密密钥

正如我们已经看到的，公开密钥和地址是从私密密钥生成的，所以私密密钥是非常重要的数据。换句话说，拥有私密密钥与拥有资产几乎是同样含义。

钱包的形式各不相同。它可以是在互联网上使用的、高度方便的 Web 钱包，可以是在 PC 上使用的桌面钱包，可以是在 USB 等终端上使用的硬件钱包，以及可以是打印在纸上的纸钱包。

此外，根据钱包的状态，可以将其大致分为连接到互联网的热钱包和脱机管理的冷钱包。

9.1.2 钱包的安全性和便利性

由于钱包中包含重要数据，其安全性自然成为关注的焦点。同时又因为其经常进行如汇款和收款这样的资产转移，它的便利性也变得非常重要。既安全又方便的钱包是大家最为期待的，但两者兼备的钱包基本上是很难实现的，如图 9.1 所示。

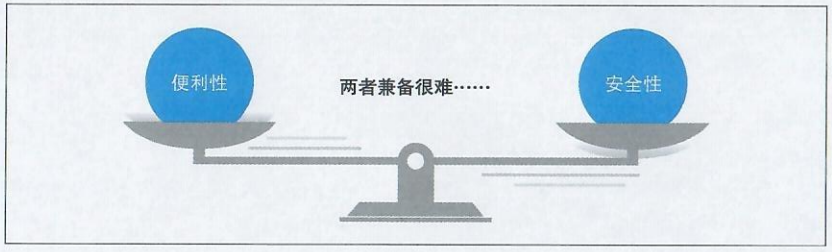

图 9.1　钱包的便利性和安全性的困境

例如，在热钱包的状态下，由于始终连接在互联网上，因此可以轻松进行诸如转账之类的资产转移。但是，它们往往容易受到经由互联网的攻击，使得其安全性相对较低。实际上，从虚拟货币交易平台泄露加密资产的事件中，大部分都是黑客攻击热钱包的事件。在冷钱包状态下，由于冷钱包是脱机管理的，因此无需担心外部黑客入侵的风险。但是，当需要进行汇款等资产移动时必须连接互联网，这使得其便利性较差。

尽管这和存储私密密钥的功能是相同的，但是由于其形式的不同，也就有了其自身的安全性和便利性的特点。读者可以根据自己的需要选择适合的钱包。

目前为止，我们已经介绍了钱包的基本内容，但最重要的一点是“钱包是用于管理私密密钥的”。根据私密密钥管理的形式以及是否与互联网连接，钱包可以分为不同的类型，每种类型都有其各自的优点和缺点，读者有必要根据不同情况正确使用它们。

但是，无论使用哪种钱包，在兼顾安全性和便利性之间的平衡时都必须处理私密密钥的复杂性，这一点是不会改变的。例如，如果将私密密钥作为冷钱包进行管理，而将公开密钥或地址作为热钱包进行管理，可以提高钱包的安全性和便利性。因此，已经设计出如确定性钱包和 HD 钱包等各种类型的钱包。

9.2 非确定性钱包和确定性钱包

根据密钥的生成方式和管理方式的不同，钱包大致可分为两类。如果将两者进行比较，会发现其在管理私密密钥和公开密钥的成本上存在显著的差异。

9.2.1 非确定性钱包

非确定性钱包是一种通过钱包中的私密密钥以一对一关系创建公开密钥的钱包如图 9.2 所示。有多少个公开密钥就需要有多少私密密钥，这使得钱包所需要管理的私密密钥的数量变多。由于是随机地生成公开密钥，它也被称为随机钱包。比特币在早期使用了这种类型的钱包。

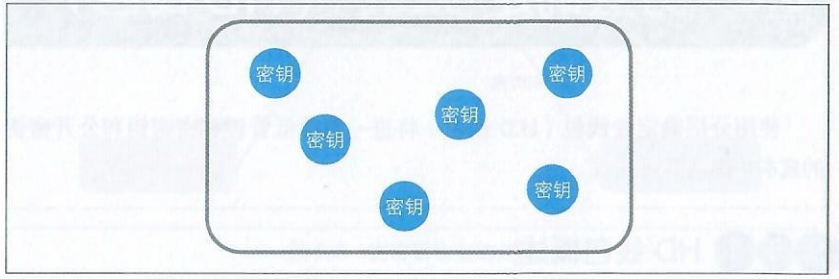

图 9.2　非确定性钱包示意图

在非确定性钱包的情况下，密钥之间没有相互关联，因此当丢失私密密钥时，没有人可以使用与之关联的公开密钥以及比特币。由于公开密钥和私密密钥是一对一关联的，一旦丢失，将很难恢复成原来的私密密钥。

9.2.2 确定性钱包

确定性钱包是指根据称为“种子”的单个随机数生成多个密钥的钱包，如图 9.3 所示。通过从作为双亲的私密密钥一个一个地生成作为子的私密密钥，在密钥之间建立了相互的依赖关系。由于存在依赖关系，甚至可以通过备份原始种子来还原所有私密密钥。

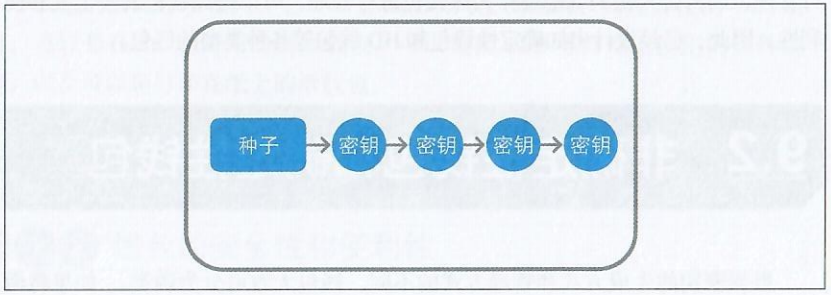

图 9.2　确定性钱包示意图

与非确定性钱包不同，确定性钱包允许在种子能够取得可靠备份的情况下还原密钥，从而轻松导出和导入钱包。由于种子可以提高钱包的便利性，因此受到了广泛关注。除此之外，确定性钱包进一步技术性地推进了分层确定性钱包（HD 钱包）的研究。

9.3 分层确定性钱包（HD 钱包）

使用分层确定性钱包（HD 钱包）将进一步降低管理私密密钥和公开密钥的成本并提高其可用性。

9.3.1 HD 钱包概述

HD 钱包与确定性钱包一样，由称为种子的随机数生成多个密钥。与确定性钱包的区别在于，它可以从一个父密钥生成多个子密钥。密钥之间的关系是树状结构，因此可以根据目的不同而分别使用，例如仅将某个分支用于特定的用途，如图 9.4 所示。另外，由于可以从父公开密钥生成子私密密钥和子公开密钥，因此不需要过于频繁地访问私密密钥，安全性也有所提高。

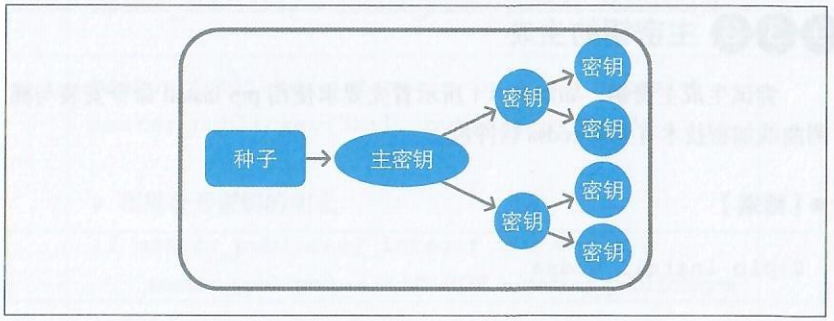

图 9.4　分层确定性钱包（HD 钱包）的示意图

9.3.2 主私密密钥、主公开密钥与链码

最开始在 HD 钱包中生成的私密密钥和公开密钥分别称为主私密密钥和主公开密钥。如图 9.5 所示显示了从随机数的种子生成主私密密钥和主公开密钥的过程。

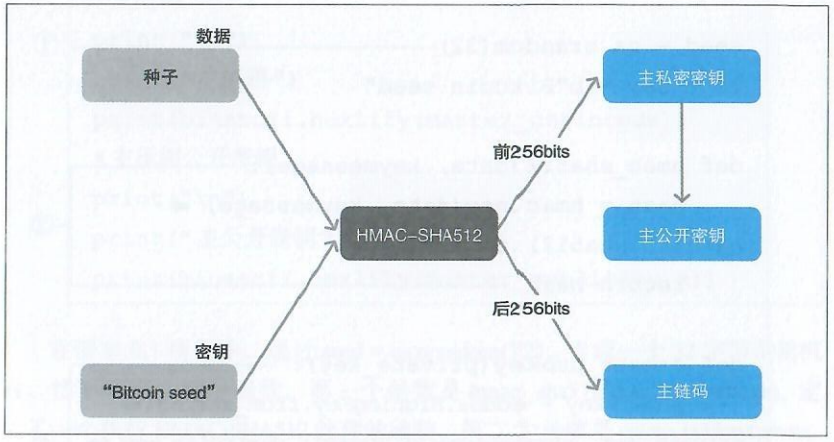

图 9.5　主密钥生成的过 程

HMAC-SHA512 是一种通过输入数据和密钥这样两个数据，输出 512 位哈希值的函数。对于比特币来说，此时的密钥固定为 Bitcoin seed（比特币种子）。输出的前 256 位是主私密密钥，后 256 位是主链码。链码是在生成子密钥时，在 HMAC-SHA512 的两个输入数据中作为密钥输入的数据。

9.3.3 主密钥的生成

尝试生成主密钥，如清单 9.1 所示首先要求使用 pip install 命令安装与椭圆曲线加密技术有关的 ecdsa 软件包。

●[终端]

```
$ pip install ecdsa
```

清单 9.1　主密钥和主链码的生成

输入

```
import os
import binascii
import ecdsa
import hmac
import hashlib

seed = os.urandom(32)                                          ①
root_key = b"Bitcoin seed"

def hmac_sha512(data, keymessage):                             ②
    hash = hmac.new(data, keymessage, ➡
hashlib.sha512).digest()
    return hash

def create_pubkey(private_key):                                ③
    publickey = ecdsa.SigningKey.from_string(➡
private_key, curve=ecdsa.SECP256k1).verifying_key.➡
to_string()
    return publickey

master = hmac_sha512(seed,root_key)                            ④
master_secretkey = master[:32]
master_chaincode = master[32:]
master_publickey = create_pubkey(master_secretkey)             ⑤

master_publickey_integer = int.from_bytes(➡                    ⑥
master_publickey[32:], byteorder="big")

# 压缩公开密钥的生成
if master_publickey_integer % 2 == 0:
    master_publickey_x = b"\x02" + master_publickey➡
[:32]
else:
    master_publickey_x = b"\x03" + master_publickey➡
[:32]

# 主私密密钥的生成
print("主私密密钥")
print(binascii.hexlify(master_secretkey))
#主链码
print("\n")
print("主链码")
print(binascii.hexlify(master_chaincode))
#主压缩公开密钥
print("\n")
print("主公开密钥")
print(binascii.hexlify(master_publickey_x))
```

在清单 9.1 所示中，通过 seed = os.urandom(32)，生成一个 32 字节的随机数。接下来定义两个函数。第一个函数是 hmac_sha512(data, keymessage)，定义了一个执行 HMAC-SHA512 处理的函数。第二个函数是 create_pubkey(private_key)，定义了通过椭圆曲线加密由私密密钥生成公开密钥的函数。这部分与第 8 章的清单 8.2 所做的处理相同。

在 master = hmac_sha512(seed,root_key) 中以 seed 作为数据并以 root_key 作为密钥来执行 HMAC-SHA512 处理。输出的 512 位（64 字节）哈希值的前半部分作为主私密密钥，后半部分作为主链码分别存储在不同的变量中。

在 master_publickey = create_pubkey(master_secretkey) 中，生成一个公开密钥，但由于这是未压缩的公开密钥，因此，在清单 9.2 所示代码中将其转换为压缩的公开密钥。因为需要根据不同的情况分别变为偶数和奇数，附加不同的前缀，所以用 if 语句将它们分别处理。第 8 章对此过程进行了说明。

清单 9.2　压缩公开密钥的生成

```
master_publickey_integer = int.from_bytes(master_publickey➡
[32:], byteorder="big")

# 压缩公开密钥的生成
if master_publickey_integer % 2 == 0:
    master_publickey_x = b"\x02" + master_publickey[:32]
else:
    master_publickey_x = b"\x03" + master_publickey[:32]
```

清单 9.3 显示了清单 9.1 的输出结果。由于输出结果是根据随机数而变化的，因此该值将在每次输出时发生变化。

清单 9.3　清单 9.1 的输出结果

输出

```
主私密密钥
b'01f2d0d4a656e128557c71d2ad6e13b378e546490f0afd8b3e0d➡
9b83f5b2a9e3'


主链码
b'92e3a2ee3081f455fb1faabc4527a8b123c17b0d34f4dc174220➡
8c6e2c1a5e62'


主公开密钥
b'0252a388d5c5d651f282abf8ea20f5f35ed7a485f26920ef1f3d➡
304285f3907c61'
```


9.3.4 子密钥的生成

生成子密钥时，将生成的链码作为密钥代入 HMAC-SHA512 中。如图 9.6 所示显示了创建子密钥（子私密密钥和子公开密钥）的过程。该过程与生成主密钥或主链码的过程非常相似，但是关键是数据生成过程和子私密密钥的生成过程。首先，作为代入到 HMAC-SHA512 中的数据，使用将索引附加到父公开密钥开始位置的数据。这样可以表示所生成的密钥是第几个密钥。其次，在生成子私密密钥时，通过将父私密密钥、索引和 HMAC-SHA512 的输出结果的前 256 位（32 字节）进行相加计算。

另外，在生成“子密钥的子密钥（孙密钥）”时，HMAC-SHA512 的输出结果的后 256 位（32 字节）作为 HMAC-SHA512 的密钥使用。通过像这样链锁式地生成密钥，可以大量地、高效地生成具有相互依赖关系的密钥。

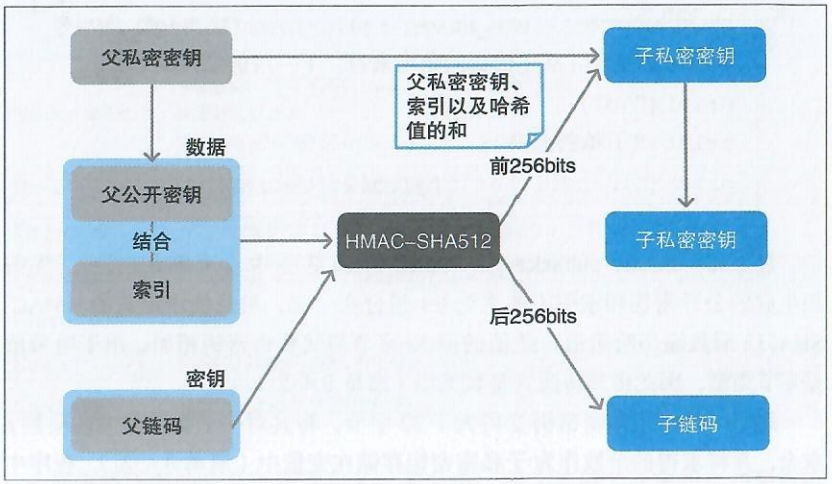

图 9.6　子密钥的生成过程

9.3.5 子私密密钥的生成

接着 9.3.3 小节的程序，如清单 9.1 所示，在其后续追加如清单 9.4 所示的程序代码。

清单 9.4　清单 9.1 中追加的代码

输入

```
(…略：清单9.1…)
index = 0
index_bytes = index.to_bytes(8, "big")
data = master_publickey_x + index_bytes ———— ①
result_hmac512 = hmac_sha512(data, master_chaincode)

# 父私密密钥和HMACSHA512结果的前半部分求和 ——┐
sum_integer = int.from_bytes(master_secretkey, "big") + \ ——②
int.from_bytes(result_hmac512[:32], "big") ——┘

p = 2**256 - 2**32 - 2**9 - 2**8 - 2**7 - 2**6 - 2**4 - 1 ——┐
child_secretkey = (sum_integer % p).to_bytes(32,"big") ——③——┘
# 子私密密钥（从主密钥的角度来看，下一层的私密钥匙）
print("\n")
print("子私密密钥")
print(binascii.hexlify(child_secretkey))
```

在 data = master_publickey_x + index_bytes 部分中（清单 9.4 ①），将前面生成的公开密钥和索引（本次为 0）组合在一起，然后使用定义的 HMAC-SHA512 函数输出哈希值。此值的前 32 字节与父私密密钥相加。由于哈希值是字节类型，因此将其转换为整数类型（清单 9.4 ②）。

此时，为防止私密密钥变得大于 32 字节，将其对一个数值很大的质数 p 求余，并将求得的余数作为子私密密钥存储在变量中（清单 9.4 ③）。程序中 p 的值是一个非常大的质数，但是这部分内容涉及高深的数学计算，这里将其省略。输出结果如清单 9.5 所示。

清单 9.5　追加清单 9.4 后清单 9.1 的输出结果

输出

```
子私密密钥
b'216b47b223d000f6fbf1804c3110aff21e2c53533f67dc0313➡
7f006ea6375cae'
```


9.3.6 扩展密钥

扩展密钥为包含私密密钥、公开密钥以及诸如链码之类的信息的密钥。之所以这么称呼它，是因为它在信息方面和功能方面比普通的私密密钥和公开密钥有所扩展。

引入扩展密钥，目的是使 HD 钱包可以在不同服务器之间移动。从在 HD 钱包中生成子密钥的过程来看，生成子密钥需要一个父密钥、一个链码和一个索引，仅有父密钥是无法生成子密钥的。因此，需要创建一个汇总了父密钥的数据以及为了生成子密钥所需要的链码之类的数据的密钥，这样不仅可以使得移植变得很方便，而且可以生成该密钥之后的子代的密钥组。扩展密钥格式如表 9.1 所示。

表 9.1　扩展密钥的格式

数据项	含　义	容　量（字节）
Version bytes	Mainnet: 公开密钥 0x0488B21E，私密密钥 0x0488ADE4 Testnet: 公开密钥 0x043587CF，私密密钥 0x04358394	4
Depth	主密钥为 0x00 时候的深度	1
Fingerprint	父公开密钥的 HASH160 的开始位置的 4 字节	4
Child number	表示是第几个子密钥的索引	4
Chain code	链码	32
Key	压缩公开密钥或私密密钥	33
Checksum	检测错误和非法操作的校验和	4

如果将压缩的公开密钥放在 Key 部分中，则为扩展公开密钥；如果将私密密钥放入其中，则为扩展私密密钥。可以从扩展公开密钥生成子公开密钥，并且不需要私密密钥。由于不需要私密密钥，因此可以在保持安全性的同时生成地址。通过利用这种特性，可以分布式地管理冷钱包和热钱包中的密钥。可以使用冷钱包管理诸如种子和主私密密钥这样的重要信息，而使用热钱包管理小型交易和高价值交易的公开密钥。

9.3.7 强化衍生密钥

强化衍生密钥是指在生成子密钥时，作为 HMAC-SHA512 数据的私密密

钥，如图 9.7 所示。在扩展密钥使用压缩的公开密钥时，公开密钥作为 HMAC-SHA512 的数据来使用。但是，这种方法具有很大的风险。让我们来思考一下使用扩展密钥在第三方服务上生成子公开密钥或地址的情况。为了生成子私密密钥，将父私密密钥和 HMAC-SHA-512 的前 32 字节加在一起。同样，使用该服务意味着父公开密钥和父链码是已知的。此时，如果子私密密钥由于某种原因泄露出去，则同一层次结构中的所有私密密钥均将能够被识别。这是因为子私密密钥只是父私密密钥和其他信息单纯的求和结果，因此使用已知的父公开密钥和父链码等，可以像如图 9.7 所示那样进行逆算求出父私密密钥。

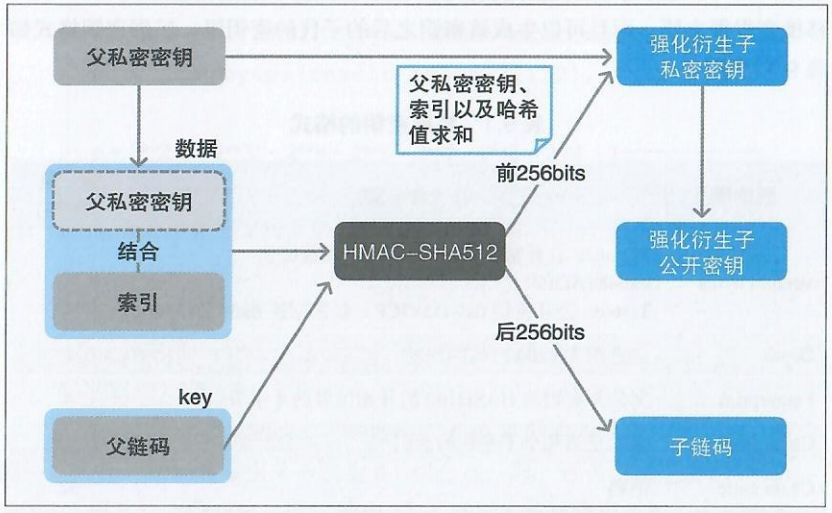

图 9.7　强化衍生密钥的生成过程

子私密密钥 ＝ 父私密密钥 ＋ 其他信息
→ 父私密密钥 ＝ 子私密密钥 － 其他信息

即使将父私密密钥保存在冷钱包中，这也可以让第三方很容易地获知。为了解决这一安全性问题，提出使用私密密钥作为数据而不是公开密钥的方案。用这种方法生成的密钥称为强化衍生密钥。

9.3.8 HD 钱包的路径

HD 钱包使用如 URL 之类的路径来识别树状结构中的密钥的位置。通过使用斜线分隔来表达密钥的世代结构，以提高可读性。此时，主私密密钥由 m 表示，主公开密钥由 M 表示。另外，一般的密钥和强化衍生密钥通过有无撇号来区分，正常密钥表示为 0，强化衍生密钥表示为 0'。通过以这种方式进行表达，使得即使层次变得很深也可以区分。如表 9.2 所示是路径表达的示例。

表 9.2　路径表达的示例

结　构	解　释
m/0	从主私密密钥生成第一代子私密密钥
M/2	第三代子公开密钥
m/0/1'	第一代子密钥的第二代的强化衍生的私密密钥

但是，如果是无限深的层次的话，管理会非常困难。因此，提出了赋予路径结构的每一层含义的方案，以便能够方便地对其进行管理。在此提案中，如下例所示，分别准备了包含五种信息的层次结构，以便于理解。

m / purpose' / coin_type' / account' /change / address_index

purpose' 代表钱包的用途，设置为 44 或 49。44 来源于 BIP44 中的提案（请参阅“说明”），用以预先定义层次结构的含义。49 是根据导入 SegWit 而设置的（请参阅“说明”）。

说明

BIP

Bitcoin Improvement Proposals 的缩写，为了改善比特币而由世界各地的工程师和研究人员提出的草案。目前为止，关于比特币技术的讨论很多，而且实际上也已经实现了很多。

bitcoin/bips

URL　https://github.com/bitcoin/bips

说明

SegWit

通过将交易数据中的签名部分保存在被称为 witness 的其他区域，可以在不改变签名正确性的情况下，避免篡改交易数据，并且可以实现压缩交易数据容量的技术。

coin_type' 设置用于指定虚拟货币的信息。比特币的主网设置为 0，比特币的测试网络设置为 1，莱特币设置为 2，以太坊设置为 60。对每种虚拟货币分配的数字由 BIP44 预先定义。还可以将新代币注册在 coin_type 中，并为该代币重新分配一个索引。

account' 对应于一个账户，可以分为个人用或组织用。另外，change 只能设置为 0 和 1，分别为收货和找零。使用 address_index 层的公开密钥生成实际的地址。表 9.3 所示是该规则的示例。

表 9.3　基于 BIP44 的路径表达式示例

结　构	解　释
M/44'/0'/0'/0/0	第一个比特币账户的第一个收货用公开密钥
M/44'/0'/1'/1/1	第二个比特币账户的第二个找零用公开密钥
M/44'/60'/1'/1/19	第二个以太坊账户的第 20 个找零用公开密钥

本章习题

问题一

请选择一个正确描述各种钱包特征的选项。

（1）冷钱包为在线管理密钥。
（2）热钱包为在线管理密钥。
（3）如果长时间存储密钥，硬件钱包丢失数据的可能性较高。

问题二

请选择一个关于非确定性钱包和确定性钱包的错误描述选项。

（1）在不确定性钱包中，管理的密钥之间不存在依赖关系。
（2）在确定性钱包中，需要管理所有生成的密钥。
（3）在确定性钱包中，只需要管理作为源头的种子。

问题三

请选择一个关于比特币 HD 钱包特征的错误描述。

（1）在 HD 钱包中，由称为种子的随机数生成多个密钥。
（2）在 HD 钱包中，一个父密钥生成一个子密钥。
（3）在 HD 钱包中，使用 HMAC-SHA512，返回 512 位的哈希值。

问题四

通过清单 9.4 中的程序进一步生成公开密钥。该公开密钥是主密钥下一级的公开密钥。