防火、防盗、防中共
——数字钱包安全三重岛链

随着$TDCCP价格的不断上涨，中共对我们的攻击一定是全方位的。那些黑客能用的手段他们一定都会，那些普通黑客不会的手段，他们也会。

在区块链世界里，安全至关重要。可以说你掌握了密码就掌握了一切，它不受法律、政府、宗教、道德、亲情等种种约束。

我把数字钱包安全防护比喻成“三重岛链”，层层保护你的数字资产，不让他人窃取。

首先我会简单介绍三重岛链是什么，他们的原理和作用是什么。然后在针对具体的安全隐患，提出一些相应的解决方案。

目录

第一岛链：助记词Mnemonic——你的安全核心

钱包的密码是一组12-24个的助记词，表示的是128位-256位的二进制数，也就是一条长达一两百位的“0”和“1”的数字串，这个是人类无法抄记的，于是有人发明了助记词，将01串变成了12-24个英文单词。

即使是12-24个单词，对人来说想要记住也是非常困难的。我们平时用的电子邮箱密码，银行卡取款密码，都是很短又好记，为何到了数字钱包这密码要这么复杂？

因为我们用了代理和中介，我们拿银行卡去ATM机取款，如果你连续三次输错密码，ATM会将你的卡吞掉。你的Gmail密码忘了，如果你反复尝超过一定次数，Google公司就会拒绝你尝试。这里的银行和Google公司都扮演了看门人的角色。

区块链是去中心化，它不像我们面对ATM、银行、Gmail这样的系统，区块链中只要你有这个链就可以反复尝试密码，谁也不能限制你反复去尝试你的密码，同样你也不能限制任何其他人去尝试。

因此区块链中要使用非常复杂的密码，避免被他人猜中，而且你还要将数字资产分散到不同的数字钱包地址中来减少风险。

如何才能保护好助记词？

下面我将从四个方面来说明如何保护助记词：

第一、防黑客攻击

黑客可能通过木马、钓鱼网站、甚至假钱包 APP 窃取你的助记词。说人话就是他人通过网络盗取了你的助记词。

解决办法：

- 使用主流的数字钱包APP，从官方下载
- 手机设备尽量不要越狱或者root
- 避免安装不可信的APP，从官方应用市场下载
- iPhone手机安全性相对优于Android手机
- 助记词抄记到现实中的本子中（避免在线）
- 助记词不要截屏、复制、粘贴

终极大招是使用冷钱包，直接隔离助记词，不让其访问网络。

第二、助记词长短

我想大家12个单词的助记词应该很难记住，那不是和24个单词的助记词有什么区别吗？反正都是记不住的。为何热钱包几乎清一色的使用12单词呢？
如果是我，我肯定会选用24个最长的，反正抄一份助记词是抄，抄两份助记词也是抄。

第三、助记词的随机性

很多人容易忽视助记词随机性的问题，而这往往正是钱包被盗的主因之一。

计算机只能处理0和1，运算过程非常确定，所以很难生成真正的随机数。为了模拟随机性，我们使用“伪随机数”。这种随机数看起来是随机的，但实际上都有一定的规律。如果有人发现了这个规律，用它生成的助记词就会变得很不安全。

有人可能会觉得像 MetaMask、Trust、Phantom等主流热钱包一定很安全，但实际上，它们在生成助记词时基本都采用了类似的伪随机数方法，这就为安全埋下了隐患。

也许你会怀疑，这么多人在用热钱包，凭什么说就有问题了，要出问题早就出了。想想打新冠疫苗的时候他们怎么说的？医生护士都打了，军队都打了，专家也说很安全，大家都打了你为什么不能打，一个道理。

比如，Ledger 等冷钱包在宣传中就会强调它们使用了某种技术生成真正的随机数，这正是在提醒你随机数的重要性。

我在这里爆个料，在中共国一些研究伪随机的专家是领国务院津贴的，伪随机数在加密领域有着非常

重要的地位。但是我们的热钱包生成助记词的逻辑都是用的类似的伪随机数。

这是否意味着所有热钱包都不安全呢？其实不然。如果你只有热钱包，采用真正随机数的助记词会更安全。后面我会介绍如何自己动手生成真正随机的助记词，然后在热钱包上使用它创建你的数字钱包账户。

第四、助记词物理安全

如果你的助记词抄在本子上，那么就会面临房子着火，遇上水灾等等不可抗拒的力量。网络上有卖钢板和字母钉子，可以将助记词印到不锈钢钢板上，这样似乎很安全。在我看来是智商税，真是有钱人不见得真有认知。

还有一种情况是助记词记录的本子被人偷了，或者CCP上门抄家，那风险也是很大。

后面的章节我会告诉你如何保存助记词，让你的助记词，不会被偷，不会被抢，甚至不会丢。

助记词的物理安全，还有一个方面，那就是冷钱包的安全。不管是热钱包还是冷钱包，它的内部都保存有一套你的助记词。如果对方拿走了你的设备，再通过非正常手段进行破解拿出里面的助记词，那也是很不安全的。

所以冷钱包也不能随便丢，优先考虑一些具有类似自毁或者自动清零机制的冷钱包，这样更加安全。

保护好你的助记词，不被偷不被抢不被猜，恭喜你已经完成第一岛链防护，进入256比特安全俱乐部。

第二岛链：助记密码（Passphrase）——你的多个分身

什么是助记密码？

助记密码（Passphrase）是给助记词再加一层保护，就像在你的钥匙上再加一道锁。即使黑客拿到了你的助记词，没有助记密码，他们还是打不开你的钱包。

助记密码的工作原理

正常情况当你的钱包生成助记词后，钱包会用一个特殊的算法（类似一个超级搓揉机），把你的原始助记词进行混合、拉伸、搅碎、重组，就像揉面团一样，一共进行2048组的反复搓揉，最终生成一个新的 512 位密码。

正常搓揉过程是没有助记密码的，为了让密码更安全，我们就在一开始将助记词和助记密码一起放到这个超级搓揉机里搅拌搓揉（其实这个助记密码还有一个专有名词——“盐”）。加入这个盐后最终会产生和没有盐情况下完全不一样的512位密码。

虽然助记密码能增加安全性，但这并不意味着你可以随意存放助记词。一旦有人拿到助记词，只要时间足够，他们是有可能破解你的助记密码的。因此，助记词的保管仍然至关重要。

助记密码的妙用

聪明的你可能已经想到——同一套助记词可以生成多个钱包！

- 无助记密码的钱包：可以用来存放少量资金，作为“诱饵”。（当饵被人动了，那么我的助记词可能就不安全了）
- 带助记密码的钱包：存放大额资产，每个助记密码都会生成一个独立的钱包。

如果你在墙内遇到CCP，非得让你交出钱包时，你可以交出没有助记密码的钱包，让对方误以为这就是全部资产。而实际上，你可能还有多个助记密码，对方根本不知道你还有多少个隐藏的钱包。

如何使用助记密码（Passphrase）

据我所知目前只有Ledger、Trezor这样的冷钱包支持，而热钱包我还没看到哪一款可以支持或者兼容的。也就是当你使用了助记密码，这个钱包必须通过Ledger或者Trezor来打开，MetaMask和Phantom以及Trust这些无法导入带助记密码的钱包。

为你的钱包设置一个易记但独特的助记密码（Passphrase），你的资产就能多一层保障，真正做到让黑客和不怀好意的人望而却步。欢迎加入512比特安全俱乐部！

第三岛链：禁用主钱包地址——化身万千

通过助记词+助记密码最终生成512位长度密码，我么也叫它种子（Seed），它可以衍生出无数个钱包地址，理论上可达 21 亿个。

无论是以太坊（Ethereum）、Polygon、Solana还是Cardano，均使用相似的派生机制来生成钱包地址。简单说是用一个固定的派生字符串+索引编号（0,1,2,3…）来生成每一个钱包地址，当然也包括对应的密码。

不同的区块链使用的不同的派生字符串，以太坊和Polygon共用一套地址，使用相同的钱包地址。Solana和Cardano则不同。虽然我们看到的地址是以不一样的，但它们是由相同的助记词+助记密码生成的。

主钱包地址是索引编号为0的地址。有些区块链支持多套派生字符串，导致在不同的数字钱包里主钱包地址也不同。但几乎所有的数字钱包，在导入新的助记词或者生成一套新的助记词后，第一个创建的钱包就是索引为0的钱包地址。

为何要废除主钱包

为什么要禁用主钱包地址，你可能从未在任何区块链的资料中看到过，不要使用主钱包地址的说法。存在是否即合理？

好吧，下面我将从一个黑客或者破解者的角度来简单跟你解释，为何一次都不要使用主钱包地址。

在区块链的交易中，我们通常能看到以下交易要素：

- 交易双方的地址
- 交易的金额
- 交易的手续费（Gas费）
- 交易的时间戳
- 有些区块链还能看到钱包地址对应的公钥

这些信息里无论如何交易双方的地址是一定会出现的。

许多人认为破解区块链钱包密码需要极大的算力，但实际上，攻击者并不一定要比拼算力，而是可以利用存储空间来实施攻击。

做为破解者，首先会生成一组助记词，再根据助记词推算出其主钱包地址，然后去整个区块链上查找有没有相同的地址，如果找到，恭喜直接破译了对方的钱包密码。破解者可以反复生成无数组助记词，以及对应的主钱包地址，把它做成一个对照表，这个表可以无限大，是不是很简单？

如果按照这个逻辑，是不是可以破解不少钱包？只要你的存储足够大，就能破解无数的钱包。事实上是做不到的，即使是12个助记词的组合，要把所有可能的组合存下来也需要5.2×10^{27} TB的存储空间。说人话，我们把地球上的每一粒沙子比做一个1T的硬盘，整个地球也只有10^{18}到10^{20}粒沙子。也就是说，需要的硬盘数量比地球上所有沙粒的数量还要多出数百万倍！正常情况下是完全不可能做到。

但我们遇到的一定是正常情况吗？我们面对一个重要的不确定因素就是伪随机数，如果伪随机数是有某种特定的规律（它一定是有规律的，设计他们的人应该是最清楚的）。那有没有可能通过伪随机缩小这个集合的可能组合，这也是一定的。再者现在的超级计算机其算力很强大，是否可以用算力换取存储空间，简单的办法就是引入一种名为彩虹表🌈的东西，这又能大大的节约存储空间。两者如果同时使用是否会成几何指数的降低存储空间？

🤔如果牛逼人物在设计伪随机生成机制时，考虑到了彩虹表的还原函数和搓揉函数，并建立了某种特殊的关联，或者针对现有的伪随机数生成规则，制作了十分特别的还原函数、搓揉函数，那就是一件细思极恐😱的事情。

一定要避免使用伪随机数生成的助记词，这是数字钱包安全的第一法则☝

至于禁用主钱包地址又是为何呢，我上面讲了对方在破解时要去整个区块链里找相同的钱包地址，如果有才算破解。主钱包地址是几乎所有钱包都会使用的，使用频率极高而且很多优秀的钱包甚至无法避开，比如Ledger、Trezor等安全度极高的钱包，因此破解者一定会以主钱包地址作为锚定来进行攻击。

说人话，主钱包地址就是攻击者破解钱包的那根引线。如果对方破解了主钱包就会顺藤摸瓜找一找有没有该主钱包的其他子钱包在使用。如果你的主钱包地址压根没有在区块链上出现过，破解者手里连个藤都没有，即使他有你的助记词，对方也不一定消耗太多算力去查找其他派生地址，所以主钱包最好一次都别用，甚至索引前几位的钱包都不要使用，这样对方要找到你需要算力和存储量都要翻N倍才可以🙏

不过，我目前还没有看到哪个数字钱包在这方面做的好，哪怕是Ledger和Trezor都做的很不到位。比如创建恢复钱包时让你输入一个你的幸运数字，之后你的钱包索引就从幸运数字开始，那这个钱包的安全度又会上升不少。（区块链的设计者在一开始就已经设计这样的扩展机制，只是无人问津）

如何使用子钱包提升安全性？

1. 禁用主钱包地址，避免成为攻击目标
主钱包地址因使用频率较高，往往成为黑客的主要攻击对象。因此，建议避免使用主钱包地址进行任何操作，直接启用一个子钱包地址作为主要存储账户。这样，主钱包地址不会出现在区块链上，从而降低被攻击的风险。

2. 每次收款使用新的子钱包地址
类似于“一次性密钥”的加密策略，每次使用新的子钱包地址收款，能够极大增强匿名性。即便别人知道某个地址与你相关，他也无法轻易追踪你的所有交易，除非通过AI分析或者掌握了助记词和助记密码。

3. 定期更换主要使用的钱包地址
建议每隔一段时间更换一次主要使用的钱包地址，以降低因量子计算攻击导致私钥泄露的风险。虽然量子计算无法直接破解助记词，但它可以针对密钥发动攻击，因此定期更换子钱包仍然是有效的安全策略。

按照以上方法操作，钱包安全性将大幅提升。剩下的，就是期待区块链技术不断升级，平稳迈入抗量子计算时代。

以上就是区块链数字钱包安全的三重岛链，接下来我会一一介绍针对三重岛链的具体安全措施。

跳出如来佛的五指山
——自制真随机助记词

看到别人用冷钱包，你是不是也很羡慕？许多人可能还在使用热钱包，而墙内的冷钱包又不一定好买。在没有冷钱包的情况下，选择一款不会采集数据，并且对本地助记词和密钥进行加密保护的热钱包，依然可以做到相对安全。

接下来我将教你如何制作真正的随机数助记词，彻底抛弃所有热钱包的伪随机数生成器

即便是冷钱包用户，如果不放心，也可以使用这个方法。不要以为使用冷钱包就绝对安全，在助记词生成上加入后门是完全不需要联网的。

整个流程分为三步

1.生成256位真随机数
2.编写一个程序，可以将随机数转换为助记词
3.在安全的离线环境中运行该程序
接下来，我会一步步带你完成整个流程。

第一步：生成256位真随机数

首先，准备 4 枚硬币（以 1 元硬币为例，正面记作 1，背面菊花记作 0）。
然后，分别投掷或者旋转4枚硬币，依此按停在桌上，例如：

0011 表示 3

1001 表示 9

1100 表示 C

上面的：0011 → 3、1001 → 9、1100 → C
转换规则参考：

二进制	十六进制	二进制	十六进制
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

每完成一组，就拿一张纸记录下对应的十六进制数字，并重复 64 组，最终会得到一个类似如下形式的字符串：
A3F8B6C1D2E4F50987AB65CDE1234789FA0BCD65432109FFEDCBA9876543210A

（如果想节省时间，可以一次投掷 8 枚硬币，这样只需重复 32 组即可；其实还有些游戏有8面的骰子，如果有几颗也是可以用的）

是不是觉得很累？其实，所谓的助记词破解，就是攻击者拿出一串与你完全相同的随机数，就算成功破解，而对方可以无数次尝试。
如果你的助记词是 12 个单词，那么只需破解 32 组即可。是不是觉得，64组似乎也不嫌多吧？😉

放心，单靠这一串 256 位的随机数并不足以破解你的钱包，对方还需要知道你的钱包地址。在钱包地址中，使用频率最高的就是主钱包地址，因此，不使用主钱包地址，也会是一种隐藏自己身份的方法。

至此，我们得到了 64 个十六进制字符，它们本质上就是24 个助记词所代表的数据。

但这些数据无法直接导入钱包，还需要按照钱包助记词的规范转换成我们熟悉的24 个助记词。

在接下来的内容中，我会告诉你如何利用ChatGPT来写一个完成这个转换的程序。

区块链和AI是灭共或者说抗衡共产党的两大利器。当你开始使用它们的时候，你会渐渐发现，共产党的灭亡不仅仅是因为自己，而是时代让它消亡！意思就是等都等死它。

第二步：编写转换程序

你可能会担心自己不会写代码，但没关系，我们让 ChatGPT 来帮忙。

1.准备工作：
准备一台电脑
或者 iPhone安装可以编辑和查看html的工具，比如 Koder Code Editor 或者 Textastic等
或者 Android安装可以编辑和查看html的工具，比如 Koder Code Editor或者QuickEdit等

2. 打开 ChatGPT，并输入以下提示词：
你是一名区块链程序员。我需要一个 HTML + JS 编写的转换程序，将 64 位十六进制字符串转换为符合 BIP39 规范的助记词。
功能要求如下：
- **提供输入框，用户可输入 64 位十六进制字符串**
- **点击“转换”按钮，输出 24 个 BIP39 助记词**
- **程序需离线运行，不依赖第三方库**
- **助记词列表从本地 ENGLISH.JS 文件加载，其中包含名为 `ENGLISH` 的助记词数组**
- **第 24 个助记词包含校验位，需计算哈希后确定。**

•计算校验值时使用 AWAIT CRYPTO.SUBTLE.DIGEST。
请帮我实现该 HTML 转换程序，谢谢！

ChatGPT会帮忙生成转换程序代码，复制并保存为DIY.html，大致结果如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>BIP39 Hex to Mnemonic</title>
    <script src="english.js"></script>
    <script>
        async function hexToMnemonic() {
            let hexInput = document.getElementById("hexInput").value.trim();
            if (hexInput.length !== 64 || !/^[0-9a-fA-F]+$/.test(hexInput)) {
                alert("请输入64位有效的十六进制字符串");
                return;
            }

            // 将16进制转换为二进制字符串（确保前导零不会丢失）
            let binStr = Array.from(hexInput.match(/.{1,2}/g), byte =>
                parseInt(byte, 16).toString(2).padStart(8, '0')
            ).join('');

            // 计算SHA-256哈希值
            let entropyBuffer = new Uint8Array(hexInput.match(/.{1,2}/g).map(byte => parseInt(byte, 16)));
            let hashBuffer = await crypto.subtle.digest('SHA-256', entropyBuffer);
            let hashArray = Array.from(new Uint8Array(hashBuffer));

            // 获取前 8 位校验位
            let checksumBits = hashArray[0].toString(2).padStart(8, '0').slice(0, 8);

            // 拼接原始二进制数据 + 校验位
            let fullBinary = binStr + checksumBits;

            // 切割成 11-bit 一组，转换为索引
            let mnemonic = [];
            for (let i = 0; i < 24; i++) {
                let index = parseInt(fullBinary.substring(i * 11, (i + 1) * 11), 2);
                mnemonic.push(english[index]);
            }

            document.getElementById("mnemonicOutput").innerText = mnemonic.join(" ");
        }
    </script>
</head>
<body>
    <h2>BIP39 Hex to Mnemonic Converter</h2>
    <label for="hexInput">请输入64位16进制字符串:</label>
    <input type="text" id="hexInput" maxlength="64" style="width: 400px;">
    <button onclick="hexToMnemonic()">转换</button>
    <h3>助记词:</h3>
    <p id="mnemonicOutput" style="font-weight: bold;"></p>
</body>
</html>
```

下载助记词列表：

下载 https://github.com/bitcoinjs/bip39/blob/master/src/wordlists/english.json
并改名为`english.js`，同时在文件开头加上`const english = `

将 english.js 和 DIY.html 放入同一目录

```
[
    "abandon",
    "ability",
    "able",
    "about",
    "above",
    "absent",
    "absorb",
    "abstract",
    "absurd",
```

原始english.json

```
const english = [
    "abandon",
    "ability",
    "able",
    "about",
    "above",
    "absent",
    "absorb",
    "abstract",
    "absurd",
```

修改后的english.js

运行转换程序

电脑版：双击DIY.html打开页面

iPhone、Android：通过编辑器打开DIY.html再预览

输入框中输入一组 64 位十六进制字符串，例如：
A3F8B6C1D2E4F50987AB65CDE1234789FA0BCD65432109FFEDCBA9876543210A

点击“转换”，最终生成的 24 个助记词如下：
physical shield race place exercise luggage burger hole social animal spin become pass trade never goddess antique youth indicate fantasy iron pave loud forum

如果转换结果和上面一致，恭喜转换程序完成了
如遇问题继续让ChatGPT帮你调整。

接下来，我们要准备一个安全的离线环境来运行
DIY.html & english.js

第三步：准备运行环境

为了避免电脑或手机被植入后门，我们必须使用干净的离线设备来进行操作

方法一：使用闲置的 iPhone

1.恢复出厂设置，抹除所有内容和设置。
2.重启后，使用非中国区 Apple ID 登录，下载 Koder Code Editor（注意：不要赋予其任何联网权限）。
3.导入文件：将 DIY.html 和 english.js 文件传输到手机。
4.断开所有网络：
•关闭手机网络、蓝牙、Wi-Fi
•若手机有 SIM 卡，取出
•开启 飞行模式，确保完全离线
5.运行转换程序
•在 Koder Code Editor 中打开 DIY.html，点击眼睛图标预览。
•输入 第一步生成的 64 个十六进制数，点击转换，生成 24 位助记词
•将助记词手写抄录在纸上
6.彻底清理设备（至关重要！）：
•在完全离线状态下，再次恢复出厂设置，抹除所有内容和设置。

•这样，即便设备上有木马，也无法导致助记词泄露。
使用闲置 Android 设备：操作逻辑与 iPhone 类似，可按照相同步骤执行。

方法二：制作U盘系统

如果家中有电脑，可以采用这种方式：
1.准备两个 U 盘：
•一个 U 盘≥16G用于制作U盘系统，这里推荐 Ubuntu
•第二个 U 盘存放 DIY.html 和 english.js 文件
（键盘建议使用有线键盘，避免蓝牙键盘带来的安全隐患）
具体操作请问ChatGPT

方法三：使用树莓派（Raspberry Pi）

如果你没有电脑，也没有闲置手机，可以考虑树莓派，这可能是最便宜的解决方案。
1.安装系统，确保树莓派环境纯净。
2.将 DIY.html 和 english.js 放入系统。
3.完全断网，通过浏览器打开 DIY.html，执行 第 6、7 步。
4.具体方法同样可以咨询 ChatGPT

生成助记词后抄录到本子上，下一章我会教你如何保存好这些助记词

不会被偷不会丢的助记词保存法

你是否曾担心你的助记词即使抄在本子上，仍然有可能遗失或被盗？如果本子被人拿走，或者因不可抗力（如火灾、浸水）损坏，那对你来说是不是也难以接受？

今天介绍一种方法，让你的助记词既不会丢，也不会被盗，即使有人拿到你的“助记词”，也无法轻易破解。

假设我们的助记词只有四个，我们以此为例：
nature sort toilet rigid

步骤一：将助记词转换为密码

找一本英文词典（或任何包含上述单词的英文书都可以），找出每个单词在书中的页码和它在该页内排第几个单词。

以英文词典为例：
nature 在第 520 页，第 10 个词条
sort 在第 748 页，第 2 个词条
toilet在第838页，第18个词条
rigid在第678页，第19个词条

依次找到所有助记词所在页码和词条序号，形成一组数据：520, 10; 748, 2; 838, 18; 678, 19

为了便于之后拆分助记词我们以6个数字表示一个助记词，不足的前面补0
520, 010, 748, 002, 838, 018, 678, 019

助记词中英文转化

有人说英文词典少见，可以尝试换成汉字

打开助记词BIP39的标准列表
https://github.com/bitcoin/bips/tree/master/bip-0039
你在这个网页english.txt找到每个英文的单词的助记词对应的行号，分别记下后
再到chinese_simplified.txt里找到相同行号对应的中文汉字
最后转成12-24个汉字就是你的中文助记词，再尝试在书本里找出这些助记词页码和位置

步骤二：加入混淆机制

为了防止别人即使拿到这串数字，也无法直接还原助记词，这是另一串密码。
让我想起了湖南卫视黄磊拷问李荣浩的那段，为什么数鸭子是24678
“门前大桥下，游过一群鸭，快来快来数一数，2、4、6、7、8。”

我们用 24678 这串数字来干扰原始编码：
第一个数字+2，第二个+4，第三个+6，第四个+7，第五个+8，然后再从头开始

520+2, 010+4, 748+6, 002+7, 838+8, 018+2, 678+4, 019+6

经过加密，新的密码变成：

52201475400984602068 2025

这串数字看似随机，即使别人拿到了，也无法直接解密（必须配合你的词典+混淆机制才可以）

步骤三：公开存放密码

可以放在email、网盘，以及各种可以公开存放的地方，哪怕是你的gettr账号备注，或者你发的一个帖子里，某一个你制作的视频里或者图片里的一串水印

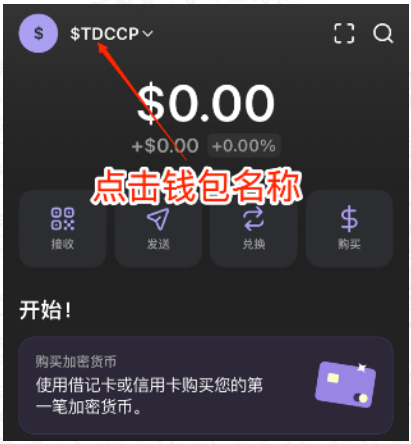
甚至用HPay给人转账的时候备注上这串码

❌不要告诉别人用的书本
❌不要告诉别人你自己的混淆规则
书本尽量找发行广泛的版本（万一书丢了呢，你得能找到一样的）

题外话：如果你理解上面整个流程，动手能力强。将公开存放的密码替换成自己记得住的简单数字串，再配合字典、混淆规则，将整个过程逆转来生成助记词，就可以设计出完全不需要公开且不会丢失的密码（这种情况生成的助记词要做随机性评估，0和1分布、转换频率、连续相同位长度、香农熵这些指标，保证其在合理范围）

Phantom如何创建子钱包

打开幽灵钱包APP首页 -> 点击左上角账户名称，进入“您的账户”列表

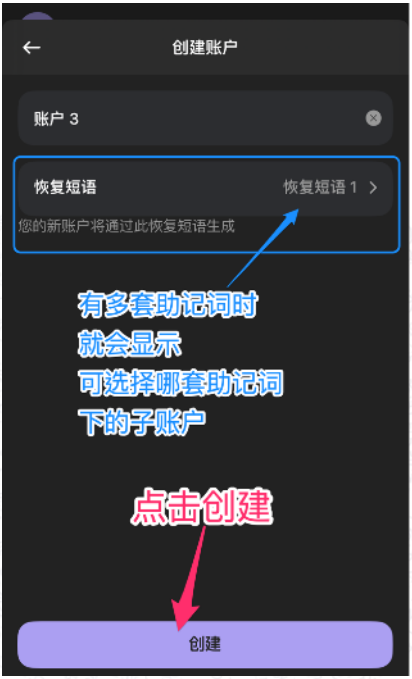


在“您的账户”中点击底部“添加/连接钱包”按钮

选择“创建新账户”

进入“创建账户”页面

- 如果你的幽灵钱包内只有一套助记词，则直接点底部“创建”即可（它们共用一套助记词）
- 如果幽灵钱包内有多套助记词，则你可以选择用哪套助记词创建子账户

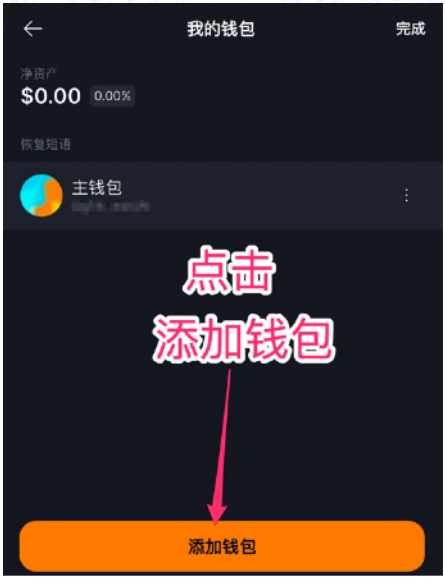


选择你要使用的子钱包账户，除了第一个自动创建的主钱包账户不要进行使用其他都可以。
（创建完其他账户后，可以点击账户编辑，移除主钱包账户。它不会真正消失，只是在钱包中不显示而已）

Solflare如何创建子钱包

打开APP首页 ->点击左上角“主钱包”->点击右上角“编辑”

这时候底部会出现一个“添加钱包”按钮，点它



对话框点击“添加钱包”

输入钱包名称随意 -> 点“添加”

子钱包功能小结

当你的子钱包账户建好后，只要切换账户转账和收款出现的二维码和钱包地址就是当前账户的。

每个助记词下面可以创建无数子账户，每个子账户都有自己的钱包地址，他人并不知道这些子账户是同一个人的（只要不相互之间转资产，一般不容易发现）。

如果你不小心删除了子钱包，没关系，你可以重新恢复她。

子钱包是有顺序的（默认从0开始创建，新的账户都是逐渐递增序号）
要恢复的钱包序号很大，你就按照新建子钱包的流程挨个加直到她出现。
如果要恢复的钱包序号是中间的或者最小的，那么你把整个钱包APP重置或者删除，再通过助记词恢复整个钱包。

当你发现恢复钱包时依然没有你的子钱包时，不要急，按照新建子钱包的流程挨个加子钱包，直到你要的她出现。

数字钱包安全小结

如果你有冷钱包一定要开启Passphrase，不要浪费冷钱包的价值；

没有冷钱包的情况下要尽量保证助记词是真随机，不要放到网络上，同时利用好子钱包账户功能，避开主钱包的使用。

关闭钱包的数据采集功能，比如Phantom钱包要关闭匿名收集，任何数据都不应让钱包采集走，哪怕这个数据看似没用，在大数据面前可能就是有用的。

如果你使用了冷钱包，为了进一步提升安全性，能自己制作助记词的话，一定要自己制作，冷钱包的助记生成有没有后门我们不知道。

冷钱包也不是绝对安全，它能隔离助记词不上网，但是不能保证冷钱包设备丢失后，被真正的高手（比如冷钱包制造商类似的人）从设备里盗取助记词，目前支持自毁的冷钱包极少。

保护好你的 $TDCCP，不被CCP抢走，这也是灭共。
扩大 $TDCCP 的使用场景，都是射向CCP的子弹。