翻译前言

这个文件由 QRL 投资者和 Tokenized Capital 创办人 Justin Poirier 委托李叔裕翻译。如果翻译版本之间存在不一致的地方，请以英文原本为正本。我们附加了增编来为读者提供在白皮书发表后关于硬币发行的变动。一个预售成功筹集了美金 416 万元的比特币和以太币。为了保留记录，我们没有改变白皮书原本的内容。

抗量子账本 (QRL)

peterwaterland@gmail.com

二零一六年十一月

摘要

私人数字金钱必须防范不断进步的计算技术来保存其长久性。在这里，我们会介绍一个密码货币账本的设计和发行。基于散列数字签名，这个帐本能抵抗经典以及量子计算攻击。

1 序言

以区块链形式记录并以工作量证明来保安的点对点互联网数值账本概念首次在 2008 年发表[11]。至今比特币仍然是最为广泛使用的密码货币。之后人们也创造了数百个类似的密码货币账本，但绝大部分都依赖同一个椭圆曲线公钥加密算法（ECDSA）生成数字签名来安全地核实交易。目前最常用的签名方法，如 ECDSA、DSA 和 RSA 等，理论上对于量子攻击都是脆弱的。我们需要探索抗量子区块链账本的设计与建造来对抗可能突如其来的非线性量子计算技术跃进。

2 比特币交易保安

目前，要从一个比特币地址使用（未使用交易输出）比特币，需要为这个比特币地址以私钥($x \in N | x < 2^{256}$)创建一个含有有效的椭圆曲线 (secp256k1) 签名。如果这个真随机生成的私钥是个不为人知的秘密或者遗失了，在合理预计内任何人就永远不能从这个地址提出任何款项。

一个特定比特币私钥的碰撞机会是 2^{256}分之一。我们可以用生日问题来估计任何比特币地址密钥碰撞的概率。要得到 0.1%的碰撞概率需要生成5.4×10^{23}个比特币地址[14]。

然而，如果这是个已签署交易。发送地址的 ECDSA 公钥就已透露并储存在区块链里。最佳实践是不重复使用地址，但截至二零一六年十一月，整个比特币账本结余的 49.58% 仍然存放在公钥外露的地址里[1]。

3 量子计算攻击向量

RSA、DSA 和 ECDSA 的保安是是基于对大整数进行因式分解的计算难度、有限域离散对数难题和椭圆曲线有限域离散对数难题的。秀尔量子算法 (1994) 可以在多项式时间内解出大整数分解难题。因此，理论上量子电脑可以从 ECDSA 公钥里把私钥重组出来。据认为，ECDSA 对量子攻击来说比 RSA 脆弱，因为前者用比较小的密钥。一个 1300 和 1600 量子位元（2^{11}）的量子电脑可以解 228 位元的 ECDSA。公开的量子电脑发展目前还没有超越2^5个量子位元或分解小整数（15 或 21）。然而，在二零一五年八月，NSA 表面上基于对量子计算的关注，弃用了椭圆曲线加密法。目前，量子计算有多先进仍然不清晰，也不能肯定任何突破会被发表，使得在互联网常用的加密协议也可以做到在后量子时代仍然是安全的。具有略为反建制的血统，比特币很可能会是量子电脑的最先对手。

如果量子计算突破是公开的，节点开发人员便可以在比特币上实施抗量子加密签名方法，并鼓励所有用家把结余从基于 ECDSA 的地址转到量子安全的地址上。要减低受影响地址的比例，在协议层面停用公钥回收是一个合理的方法。这样的有计划升级也可能会导致中本聪拥有的一百万比特币流动及相应的价格波动。

一个较差的场景是出现了不公开的非线性量子计算技术跃进，然后发生针对公钥外露的比特币地址的微妙量子攻击。当公众得悉这种盗窃的规模时，强大的抛售压力以及对系统的信心尽失会对比特币汇率产生破坏性影响。比特币作为价值储藏（“数字黄金”）的地位将会受到极大损害，对世界的影响也会极其严重。在

这方面，作者认为应该在密码货币账本上测试抗量子加密签名，并且也许可以创建一个用来应付黑天鹅事件的备用价值储藏。

4 抗量子签名

现今有数个被认为是抗量子的加密系统：基于散列加密、基于编码加密、基于格加密、多元二次方程加密和密钥加密。这些加密方法都被认为在足够长的密钥下可以抵抗经典与量子计算攻击。

现在已有前向安全基于散列的数字签名方法。安全要求很低，这些方法的安全要求很低，只依赖一个加密散列函数的抗碰撞属性。改变散列函数就会生成另一个基于散列数字签名方法。基于散列的数字签名已经过充分研究，是未来后量子签名的主要方式。因此，它也被选为 QRL 的后量子签名类。

5 基于散列的数字签名

抗量子基于散列的签名依赖一个单向加密散列函数。这个函数输入一个消息m，输出一个固定长度n的散列摘要h, 例如 SHA-256、SHA-512。在加密散列函数的应用下，用暴力破解法去从h求m（抗原像攻击）或从h_2求h，$h_2 = hash(h)$（抗第二原像攻击）都是计算上不可行的。此外，求两个输出相同的h的消息$(m1 \neq m2)$也是非常困难的（抗碰撞）。

Grover 演算法可以用$O(2^{n/2})$个运算步骤来尝试寻找散列碰撞或者进行原像攻击来求m。因此，要保护 128 位安全，散列摘要长度n至少要是 256 位元，假设加密散列函数是完美的。

基于散列的数字签名需要一个公钥pk来核实和一个私钥sk来签名消息。下面我们会讨论在区块链账本里用不同基于散列的一次性签名（OTS）的适合性。

5.1 Lamport-Diffie OTS

在 1979 年，Lamport 描述了一个长度为m位元的消息（通常是抗碰撞散列函数的输出）的基于散列的一次性签名。密钥对生成的结果是m对随机密钥$sk_j^m \in \{0,1\}^n$，其中$j \in \{0,1\}$，即私钥是：$sk = ((sk_0^1, sk_1^1, \dots, \dots, (sk_0^m, sk_1^m))$。设$f$为单项散列函数$\{0,1\}^n \to \{0,1\}^n$，并且生成$m$对公钥$pk_j^m = f(sk_j^m)$，即公钥是：$pk = ((pk_1^0, pk_1^1), \dots, \dots, (pk_0^m, pk_1^m))$。签名就是按位检视消息的散列来选择$sk_j$（即如果*位元* $= 0$，$sk_j = sk_0$，*位元* $= 1$，$sk_j = sk_1$），生成签名：$s = (sk_j^1, \dots, sk_j^m)$，披露了私钥的一半。核实签名就是按位$(j \in \{0,1\})$检视消息的散列来核实$\left(pk_j = f\left(sk_j\right)\right)^m$。

假设按 Grover 演算法，我们需要 128 位安全，消息长度是 SHA256，$m = 256$ 和 $n = 256$的固定散列输出，每个使用的 OTS 就得到一个$pk = sk =$ 16kb 和一个 8kb 的签名。一个 Lamport 签名应该只使用一次，并且可以很快生成，但缺点是密钥大、签名大，即是交易也大，所以作为公开区块链账本是不实际的。

5.2 Winternitz OTS

对于一个长度为 m 位元的消息摘要M，具有长度为n位元的密钥和公钥，一个单向函数$f: \{0,1\}^n \to \{0,1\}^n$和一个 Winternitz 参数$w \in N | w > 1$，Winternitz 一次性签名的一般概念是在一系列随机密钥$sk \in \{0,1\}^n, sk = \left(sk_1, \dots, sk_{m/w}\right)$上使用迭代散列函数，生成长度为$w - 1$的散列链，末端是公钥$(pk \in N | \{0,1\}^n), pk^x = f^{2w-1}(sk_x), pk = \left(pk_{1,\dots,} pk_{m/w}\right)$。

不同于 Lamport 签名的按位检视消息摘要，在这里我们将消息逐次解析，每次w位元，抽出数$\mathrm{i} \in \mathrm{N}, \mathrm{i} < 2^w - 1$，并从这个数生成签名 $s_x = f^i(sk_x)$，$s = (s_{1,\dots,} s_{m/w})$。我们可以将$w$加大来达到较小密钥/较小签名与计算工作量增加之间的平衡[10]。

核实就是从M和s生成$pk_x = f^{2^{w-1}-i}(s_x)$，并确实公钥相符。

用 SHA-2 (SHA-256)作为单向加密散列函数$f: m = 256$，$n = 256$，$w = 8$，得到$pk = sk = s$ 的大小为 $\frac{(m/w)\mathrm{n}}{8}$ 字节 = 1kb。

生成pk需要f^i个散列迭代，其中$i = \frac{m}{w}2^{w-1} = 8160$每生成一对 OTS 密钥。当$w = 16$时，密钥和签名的大小减小一半，但$i = 1048560$就不实际了。

5.3 Winternitz OTS+ (W-OTS+)

Buchmann 将原来的 Winternitz OTS 改变为把单向迭代函数重复使用在随机数x上，并设置参数密钥k。这参数是从上一次$f_k(x)$的迭代生成的。如果使用了伪随机函数（PRF），在自适应选择消息攻击下这是非常难伪造的。对于既定的参数，我们并且可以为其计算一个安全证明[3]。以在函数上随机游走来代替迭代，它并不需要抗碰撞散列函数族。Huelsing 再提出了一个 W-OTS+变式，以在迭代链函数上增加一个位屏蔽 XOR 来生成具有相同位元安全度但较小的签名[6]。W-OTS(2011 变式)/ W-OTS+与 W-OTS 的另一个分别是消息每次以$log_2(w)$，而不是w，个位元解析，减少了散列函数的迭代但增大了密钥和签名的大小。

我们现在为 W-OTS+做一个简略描述。以安全参数$n \in N$，相应于消息长度(m)，由选定的加密散列函数和 Winternitz 参数$w \in N | w > 1$（通常是 $\{4, 16\}$）决定的以位元为单位的密钥和签名，我们计算l。l 是在一个 WOTS+密钥或签名里n位元长的串元素的个数，$l = l_1 + l_2$：

$$l_1 = \lceil \frac{m}{log_2(w)} \rceil,\ l_2 = \lfloor \frac{log_2(l_1(w-1))}{log_2(w)} \rfloor + 1$$

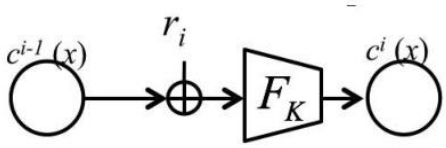
我们使用一个带有密钥的散列函数 $f_k : \{0,1\}^{\mathrm{n}} \rightarrow \{0,1\}^{\mathrm{n}} \mid \mathrm{k} \in \{0,1\}^{\mathrm{n}}$。以伪代码显示：

$$f_k(M) = Hash(Pad(K)||Pad(M))$$

其中 $Pad(x) = \left(x||10^{b|x|1}\right), |x| < b$

链接函数$c_k^i(x, r)$：输入为$x \in \{0,1\}^n$，迭代计数值 i，密钥$k \in K$，随机化元素$r = \left(r_1, \ldots, r_j\right) \in \{0,1\}^{n*j}$，$\mathrm{j} \geq \mathrm{i}$，定义是：

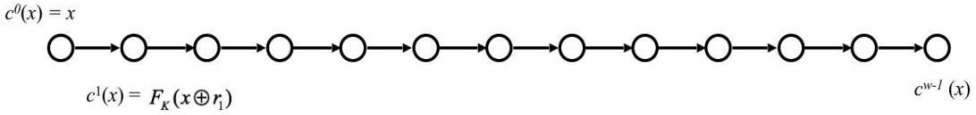
图 1、W-OTS+ 链接函数



其中：

$$c^i(x, r) = \begin{cases} x & 若\ i = 0; \\ \left(c_k^{i-1}(x, r) \oplus r_i\right) & 若\ i > 0; \end{cases}$$

图 2、散列链生成例子



这就是把上一个c_k的迭代与随机元素的按位 xor 输入到f_k，然后再把结果输入到c_k的下一个迭代。

5.3.1 签名密钥

要建立密钥sk，(用 PRF)随机均匀选择$l+w-1$个n位符串，以头l个作为密钥$sk=(sk_1,\dots,sk_l)$，余下的$w-1$个n位符串成为$r=(r_1,\dots,r_{w-1})$。随机均匀选择一个函数密钥k。

5.3.2 核实密钥

公钥是：

$$pk=(pk_0,pk_1,\dots,pk_l)=\left((r,k),c_k^{w-1}(sk_1,r),c_k^{w-1}(sk_2,r),\dots,c_k^{w-1}(sk_l,r)\right)$$

注意pk_0包含r和k。

5.3.3 签名

签名步骤：解析长度为m的消息M，使得$M=\left(M_1,\dots,M_{l_1}\right),M_i\in\{0,w-1\}$（建立$M$以基数为$w$的一个表现）。

然后计算并追加长度为l_2的校验值C：

$$C=\sum_1^{l_1}(w-1-M_i)$$

使得：$M+C=b=(b_0,\dots,b_l)$

签名是：

$$s=(s_1,\dots,s_l)=\left(c_k^{b_i}(sk_1,r),\dots,c_k^{b_l}(sk_l,r)\right)$$

5.3.4 核实

要核实签名，从M重组$b=(b_1,\dots b_l)$。

如果$pk=\left(c_k^{w-1-b_1}(s_1),\dots,c_k^{w-1-b_l}(s_l)\right)$，签名就是有效的。

W-OTS+的安全度至少是$n-w-1-2\log(lw)$个位元[3]。一个正常的签名，$w=16$，使用 SHA-256 ($n=m=256$)是ln个位元或 2.1kb。

6 Merkle 树签名方法

虽然一次性签名能为交易签名和核实提供满意的加密安全，它的主要缺点是只能安全地使用一次。如果账本地址是基于只是一个 OTS 密钥对的公钥的转换，区块链账本就会受到极大的限制，每一个交易都需要移动发送地址的全部款项，不然款项就会有被盗窃的风险。

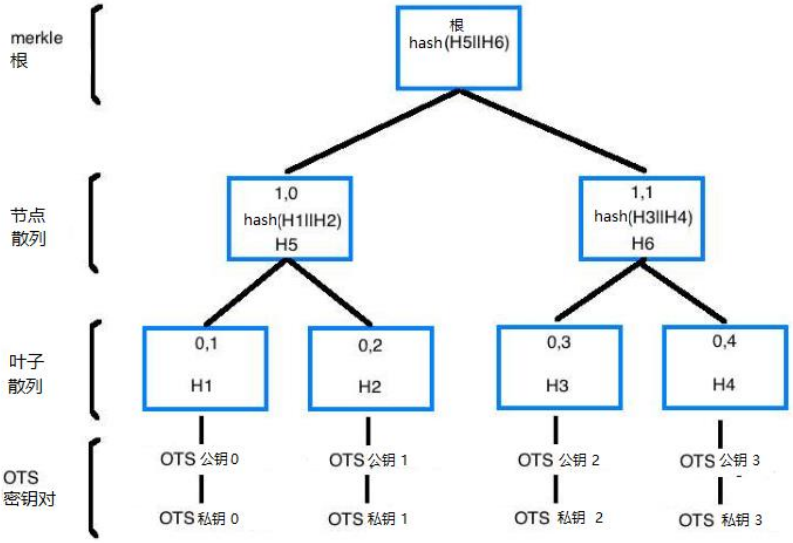
一个解决方法是在签名方法里为每个账本地址加进多于一个有效的 OTS 签名，容许签名的个数可以与预设的 OTS 密钥对一样多。一个名为 Merkle 树的二叉散列树是实现这个目标的合理方法。

6.1 二叉散列树

Merkle 树的背后概念是一棵倒置的树。树的每个父节点是将其所有子节点连接后的散列。这样逐层向上计算到树根，就形成一棵倒置的树。我们可以计算根来加密证明每个节点或叶子的存在。

一棵 merkle 树由n个基叶组成，到 merkle 根的高度为h ($n=2^h$)——从叶散列（层 0）开始逐层向上数。在这个假设账本用例中，每个叶节点是一个随机预设的 OTS 公钥的散列值。在下面的树中，每对叶散列上面的节点是其子散列的连接的散列值。

图三、Merkle 树签名方法例子



逐层向上重复这个运算，直到在树的根散列——merkle 根——汇合。

在上图中的例子，以 merkle 根为公钥，我们可以用四对预设的 OTS 密钥来生成四个加密安全的有效一次性签名。二叉散列树的 merkle 根可以转换为账本地址（可能用追加校验值迭代散列来做）。消息M对于既定的 OTS 密钥对的完全签名S包含：签名s、OTS 密钥号n和 merkle 认证途径，对于 OTS·密钥对 0（n=0）就是：

$$S = s, n, OTS\ 公钥\ 0, H1, H2, H5, H6, 根$$

OTS 公钥和叶散列值都可以从s推算出来，并且父节点也可以用其子节点来计算，所以这也可以缩短为：

$$S = s, n, H2, H6, 根$$

若从s和m核实 OTS 公钥而证明了S是有效的，则检查 merkle 认证途径的散列值可以重建对称的 merkle 根（公钥）。

6.2 状态

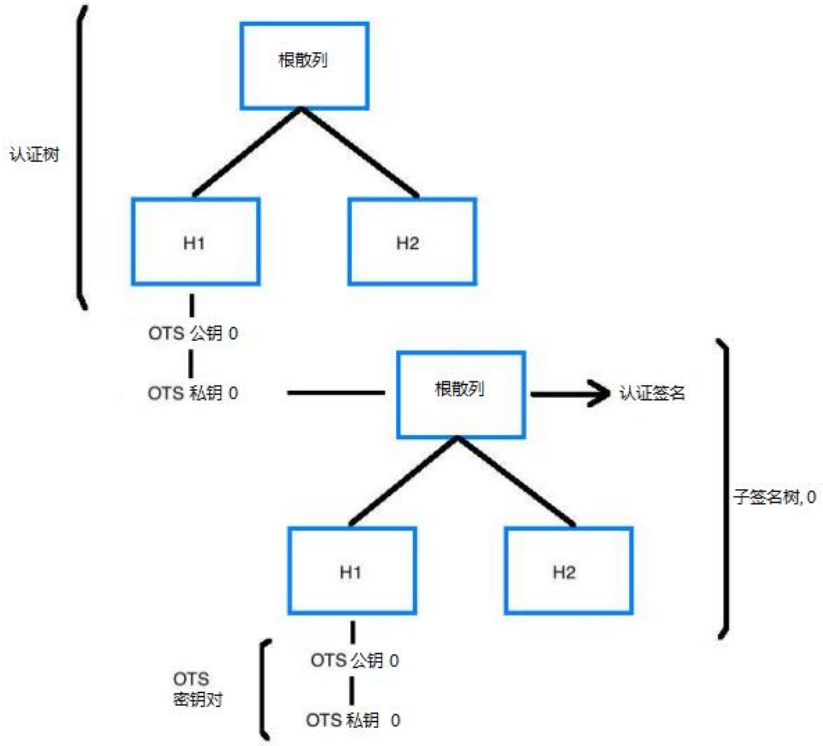
上述的 merkle 签名方法（MSS）依赖不重复使用 OTS 密钥，因此它与签名的状态或正在被记录的签名交易的状态有关。通常在实际应用中这可能会是一个问题，但一个不可变的公开区块链账本是状态加密签名方法的理想存储介质。在 2015 年发表的 SPHINCS 是一个新的基于散列的加密签名方法，它具有实际可用的无状态签名和2^{128}位安全保障[2]。

6.3 超树

基本 MSS 的一个问题是可能签名的个数是有限的，并且在计算 merkle 树之前就要预先生成所有的 OTS 密钥对。密钥的生成时间与签名时间随着树的高度h按指数级增长，这意味着生成大于 256 个 OTS 密钥对的树对时间和计算量的需求都非常庞大。

要延迟在生成密钥和树时的运算，并且增多可用的 OTS 密钥对，我们可以使用以 merkle 树组成的超树。这个策略的一般概念是用父 merkle 树——称为认证树——的叶散列的 OTS 密钥来签名子树的 merkle 根。

图四、链接 merkle 树



最简单的认证树（高度$h = 2$）是以2^1个 OTS 密钥对预先计算的。当需要第一个签名时，我们计算一棵新的 merkle 树（签名树 0），并用一个认证树 OTS 密钥对签名。签名树由n个叶散列和其对应的 OTS 密钥对组成，用来签名消息。当签名树的每一个 OTS 密钥对都已被使用后，我们就会用认证树的第二个 OTS 密钥对来签名下一个签名树（签名树 1）来准备下一批签名。

这样的超树的签名S略为复杂，它包括：

1. 签名树的：$s, n, merkle$途径, 根

2. 每棵认证树的：（子树 merkle 根的）$s, n, merkle$途径, 根

理论上，我们可以从认证树向下嵌套一层层的树来无穷伸延原来的 MSS。签名大小随着签名每一棵新树线性增大，而超树的签名量则按指数级增长。

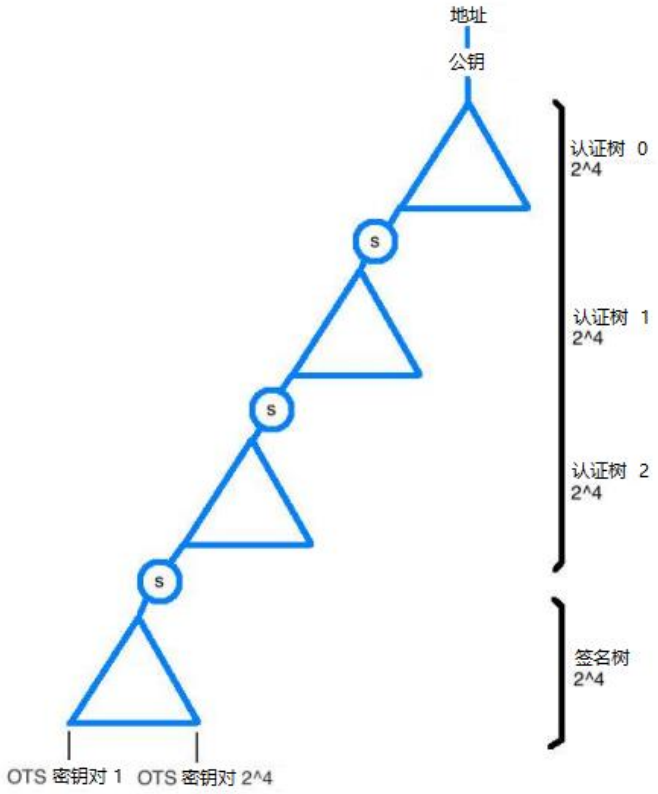
6.3.1 超树例子

我们现在用一个例子来示范怎样容易地用超树来伸延 MSS。设有初始认证树，高度$h_1 = 5$，有2^5个叶散列和相关的 OTS 密钥对。我们转换这棵树的 merkle 根来生成一个账本地址，并且建立另一棵相同大小（$h_2 = 5, 2^5$个叶散列和 OTS 密钥对）的 merkle 树作为签名树。这样我们可以用了 32 个签名后才再需要建立下一棵树。可能签名的总数是$2^{h_1+h_2}$，在这个例子是$2^{10} = 1024$。

以下是在 Macbook pro 2.7GHz，i5 处理器，8gb 内存上建立 OTS 密钥对和不同大小的 merkle 认证树的数据（未优化 python 代码，Winternitz OTS）：$2^4 = 0.5s, 2^5 = 1.2s, 2^6 = 3.5s, 2^8 = 15.5s$。由初始生成的两棵$2^4$树组成的超树用了大约$1s$，而相同签名量的标准$2^8$MSS 树则需要$15.5s$。

增加超树的深度（高度）延续这个趋势。由四个链接的2^4认证树和2^4大的签名树组成的超树可以提供$2^{20} = 1{,}048{,}576$个签名。这棵超树的签名大了，但只需要$2.5s$的时间来建立。

图五、超树结构

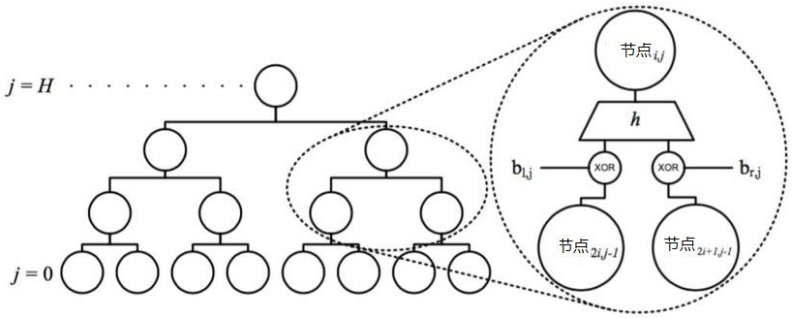


超树并不需要是对称的，所以如果它初始时是由两棵树组成，之后也可以用签名更多层的树来延伸。因此，账本地址的签名开始是小的，但会随着超树的深度增大。使用 merkle 超树来建立和签名账本地址的交易永远不会可能需要$> 2^{12}$个交易。因此，可以用深度$h = 5$的超树和轻量的计算来做2^{20}次安全签名是非常足够的。

6.4 XMSS

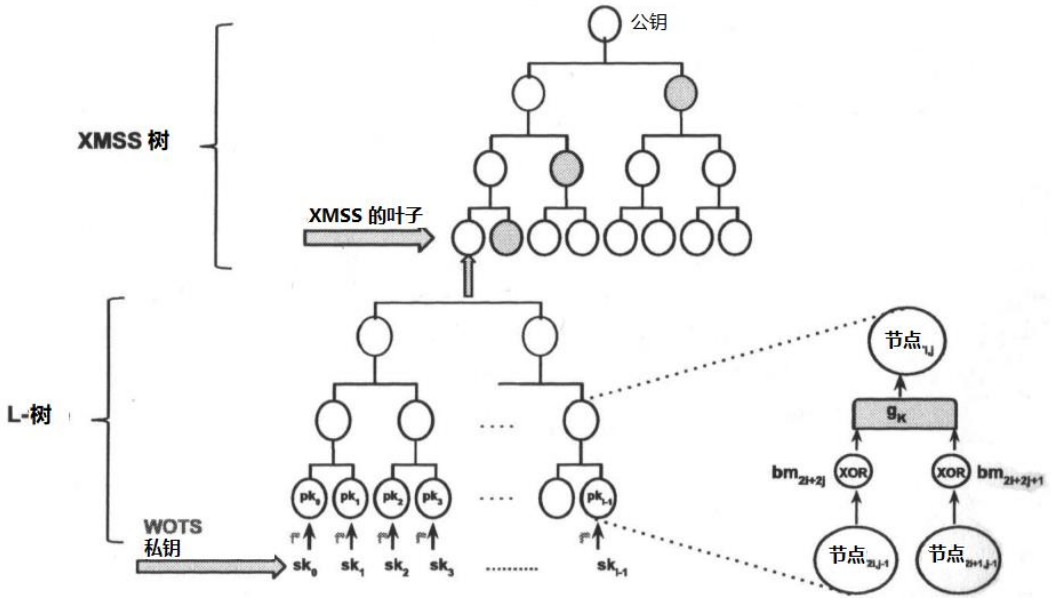
2011 年 Buchmann 等人报道了延伸 merkle 签名方法（XMSS），这方法也在去年发表为互联网工程专责小组（IETF）草案[4][7]。在既定的消息攻击下，这方法可以被证明为存在性不可伪造和前向安全，并且只需要低度的安全要求：PRF 和抗第二原像攻击散列函数。它容许用 merkle 树伸延一次性签名，主要分别是在子节点上进行位屏蔽 XOR 后才把散列连接到父节点。使用位屏蔽 XOR 容许我们替换抗碰撞散列函数。

图一、XMSS 树结构



树的叶子也不是 OTS 密钥对散列，而是子 L-树的根，包含 OTS 公钥，由l个组成底部的叶子。一次性签名使用 Winternitz OTS+（但最初描述的是 2011 年的变式）。

图七、XMSS 结构[8]



XMSS 公钥的位长是$(2(H + [logl]) + 1)n$，XMSS 签名的长度是$(l + H)n$，XMSS 签名密钥的长度$< 2n$。

Buchmann 用以下的设置做了测试：Intel(R) i5 处理器，2.5 Ghz，高度$h = 20$的 XMSS 树，$w = 16$，加密散列函数是 SHA-256（$m = 256$），上至大约一百万个签名。结果是签名用了 7ms、核实用了 0.52ms、生成密钥用了 456 秒。以公钥大小为 1.7kb、私钥 280 位、签名 2.8kb，达到的安全度是 196 位。XMSS 是个不错的方法，其主要的缺点是极长的密钥生成时间。

6.5 XMSS 树效能

用未优化的 python 代码在上述的硬件（ Macbook pro 2.7Ghz i5 处理器，8gb 内存）处理一个有 4096 个叶子的 XMSS 树（$h = 12$）的 QRL 测试节点组织，并且所有的密钥及位屏蔽从基于散列的 PRF 生成，结果是 32s。这包括用 PRF 生成多于 8000 个位屏蔽和多于 300,000 个 sk 片段。一个更高效率的 merkle 树遍历算法并且只需要为每个密钥链函数计算$w - 1$个散列是效能大幅度高于传统 MSS 的原因。

这个设置组合（11.75kb 十六进制符串编码）达到了 5.75kb 的完整签名大小，包括：OTS 密钥对n、签名、XMSS 认证途径、OTS 公钥和 XMSS 树公钥（包括 PRF 公钥种子和 XMSS 树根）。

以 PRF 及从一个随机种子生成的树（具有不同的签名量）的效能测试结果是：(h=9) 512 4.2s, (h=10) 1024 8.2s, (h=11) 2048 16.02s。

7 签名方法建议

7.1 安全要求

QRL 的设计的一个重点是签名方法的加密安全能防御现今及未来几十年的经典和量子计算攻击。基于 SHA-256，$w = 16$的 XMSS 可以提供 196 位安全，预测可以抵御暴力计算攻击到 2164 年[9]。

7.2 QRL 签名

我们的建议是一个由链接的 XMSS 树组成的可伸延状态非对称超树方法。这个方法的优点是使用已核实的签名方法，并且容许生成可以签名交易的账本地址，避免了庞大 XMSS 结构的预先计算延迟。基于安全及效能考虑，我们选择了 W-OTS+为基于散列的一次性签名。

7.3 超树结构

7.3.1 密钥与签名的大小

密钥与签名的大小随着超树里树的个数线性增加，但签名量则按指数级增长。从 XMSS 树导出的公钥和签名的大小（基于 2011 年的描述），其中$w = 16$，$m = 256$，h 是树高，加密散列算法为 SHA-256，是：

• $h = 2, 2^2$个签名：公钥 0.59kb，签名 2.12kb (0.4s)

• $h = 5, 2^5$个签名：公钥 0.78kb，签名 2.21kb (0.6s)

• $h = 12, 2^{12}$个签名：公钥 1.23kb，签名 2.43kb (32s)

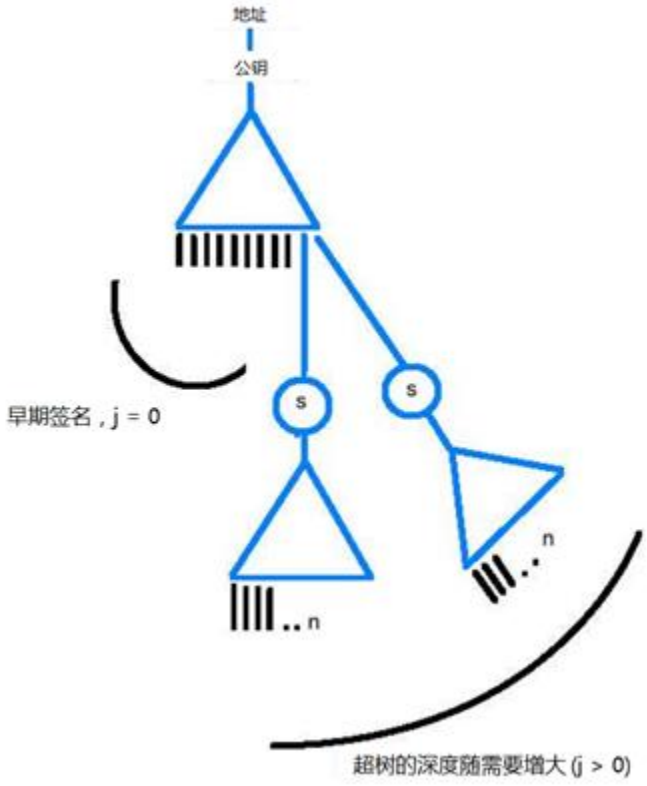
• $h = 20, 2^{20}$个签名：公钥 1.7kb，签名 2.69kb (466s[3])

考虑到可以在 3s 内建立一棵终极签名量为2^{20}而签名大小为 8.84kb 的 XMSS 超树，与签名大小为 2.69kb 但需要用 466s 来建立之间的平衡，前者应该是可以接受的。

7.3.2 非对称

建立非对称的树容许早期的签名在同一个 XMSS 树内产生，但需要伸延这棵树来处理后来的签名，削减了整体签名量。但是，这不会对区块链账本有负面影响，钱包也可以给予用户签名量与签名/密钥大小之间的选择。在所有情况下，树的深度大至$j = 2$就应该足够了。

图八、非对称超树



7.4 QRL 超树规格

以下是标准超树结构的默认参数：

• $j = 0 (j \in \{0 \leq x \leq 2\}), h = 12$ $(h \in \{1 \leq x \leq 14\})$，可能签名个数的上界：$2^{36}$，最小签名：2.21kb，最大签名：7.65kb。

就是，一棵$h = 12$，可以提供 4096 个签名的 XMSS 树，如需要的活，可以用更多的树伸延到$h = 14$。实际上，大多数用户都不会有这个需要。

7.4.1 QRL 签名例子

假设最复杂的超树结构（$j = 2$，$h = 14$，交易消息签名m，n为每棵 XMSS 树的 OTS 密钥对的位置）需要：

• 签名树，$j = 2$：m的 OTS 签名、n、merkle 认证证明、签名树的 merkle 根

• 认证树，$j = 1$：签名树（$j = 2$）的 merkle 根的 OTS 签名、n、merkle 认证证明、merkle 根

• 原本的 XMSS 树，$j = 0$： merkle 根（$j = 1$）的 OTS 签名、merkle 认证证明、merkle 根

核实的步骤是首先从m和签名生成 OTS 公钥，然后确实提供的 merkle 认证证明可以生成签名树的 merkle 根。用这个作为下一个 OTS 签名的消息，从而生成下一个 OTS 公钥，用提供的 merkle 认证证明重建认证树的 merkle 根来成为下一个认证树 OTS 签名的消息，等等。一个签名只有在最高的树——即是原本的 XMSS 树（$j = 0$）——的 merkle 根正确地生成时才是有效的。

注意核实 XMSS 树签名是不需要 OTS 公钥的。实际上，我们可以导出每棵树的 merkle 根，所以——如果发送账本地址是已知的——在超树签名核实时也可以省略它（因为这是从在 QRL 签名内最高的 XMSS 认证树（$j = 0$）的 merkle 根计算导出的——见下文"账户"段）。

这是个状态签名方法，所以钱包必须在超树内为既定地址生成一个 XMSS 树时保留和更新n。

7.5 PRF

使用种子的 PRF。散列运算消息认证码、确定性随机位元发生器（HMAC DRBG）。

7.6 确定性钱包

我们可以用一个种子来生成很大的 XMSS 树来在一段长时间内满足大部分用户。种子从一个安全的熵源产生，然后把种子以安全 PRF 函数生成一组伪随机密钥来建立一棵树。使用同一棵 XMSS 树的缺点是用户只能使用一个地址（虽然披露公钥并不是 MSS 的问题）。

一个比特币或以太坊地址是从相应的公钥导出的，所以一个私钥或公钥只能建立一个地址。然而，XMSS 地址却是从含有 merkle 根及公开种子的公钥 PK 导出的。如果种子保持不变而 OTS 密钥对的个数可以改变，每个改变也会改变了 merkle 根。因此，每添加或减除一个 OTS 密钥对都会改变导出的地址。

钱包/节点软件可以用这个特性来生成许多 XMSS 树的变式（基于需要用同一个初始种子伸延或收缩 XMSS 树）来按需要建立任何数目的独特地址。安全、简洁及有状态地记录这个信息所需要的计算量也是微不足道的。

8 密码货币设计参数

我们会在这白皮书余下的章节列出 QRL 账本的设计参数建议。账本的重点是一个高度安全、可以抵御经典和量子计算攻击向量的公开区块链。这白皮书是第一稿，所以每一方面将来都可能会更改。

8.1 权益证明

QRL 是一个以权益证明算法来保障安全的公开区块链账本。一个 epoch（以下翻译为“纪元”）是 10,000 个区块。权益证明者是从上一个纪元的权益交易决定的。一般概念是每个权益证明者签名一个交易，这个交易含有一个 10,000 散列长的迭代链的末端散列（每个迭代可能会用一个位屏蔽 XOR 来减低散列函数的安全要求）。当在链内确实了权益交易后，网络的每一个节点就可以把权益地址的加密身份捆绑到下个纪元的散列链上。

8.1.1 设计与随机性

对于每一个区块，每个在当前纪元证明权益的证明节点显露链的上一个相连散列来加密证明参与及投票其为成功的区块选择者。

我们用 HMAC DRBG 从区块链来生成一个 32 位的伪随机数列。种子是从区块链得来的（最初用创世块，之后每个纪元都添加熵，而熵是从最近的区块头散列的连接取得的）。

因此，若一个权益证明者显露的散列值离一个区块的 PRF 输出最接近，他就会被选为区块选择者。在这个设计中，作弊是非常困难的，因为在建立散列链的一刻，权益证明者并不知道 PRF。并且，迭代（有密钥的）散列链基本上是个随机数列。最后，就算权益证明者互相勾结，他们也不可能知道其他权益证明者的散列链内容，因为内容还未显露。

要防止区块截留攻击，如果权益证明者不能在提交有效的显露散列后出示有效的区块，这会被视为整个从那个地址来的区块已经遗失，他在一个惩罚期间内会被禁止参与。

为了缓解女巫节点或低结余权益证明者地址发动的空区块攻击，我们使用一个有弹性的权益证明者列表门槛。区块奖励是以基于权益地址的结余加权支付的。除了显露散列消息外，每个节点也在其交易池内展示一个 txhash 排序列表的 merkle 树根散列和在等待区块的交易的个数。每个节点从顶部和底部切出一部分来估计下一个区块的的期待交易个数。如果区块是空的或少于期待，每个区块的允许权益证明者个数就会向上收缩（从贫穷到富有次序排除权益证明者）。相反，如果所有的区块选择者都是诚实的，允许的权益证明者个数就会增加。在受权益证明的纪元内，款项不能移动，这样就能防止女巫攻击以建立从多权益证明者地址来尝试操纵区块选择。

8.2 收费

相对于其他账本，这个账本的交易比较大。这意味着需要为每个交易征收交易费。作者认为人为的收费市场是不需要的，也与公开区块账本的理念背道而驰。每个已支付最小费用的交易应该和其他任何交易一样有效。挖矿者愿意接受的最小费用应该是浮动的，并由市场制定。就是说，节点和挖矿者通过竞争来自行决定费用的下限，但在协议层面会有一个绝对最低费用。因此，挖矿者会自行要求把在内存池的一些交易包括在一个区块里。

8.3 区块

8.3.1 区块时间

区块之间的时间在比特币大约是 10 分钟，但基于自然变动，有时挖掘下一个区块可能要等一段相当长的时间。比较新的账本设计（像以太坊）对此有所改善，区块时间短至 15 秒，而安全度没有降低，也没有从过多孤儿块或陈旧块引致挖矿者高度集中。以太坊使用一个幽灵协议（GHOST protocol）的修改版本，容许陈旧块或孤儿块也可以被包括在区块链里并得到奖励[13, 5]。

因为 QRL 一开始就用权益证明，我们预计可以安全地以 15 到 30 秒作为区块时间。

8.3.2 区块奖励

每个新区块会含有一个初始"coinbase"交易。这个交易包括一个挖矿地址。这个地址会收到一个奖励。这个奖励的值是 coinbase 奖励和区块里所有交易费的和。

区块奖励是基于被选为区块选择者的权益证明者地址的结余加权的。

挖矿节点会为每个区块重新计算区块奖励，并跟随硬币发行时间表。

8.3.3 区块大小

为了避免争议，我们会使用一个开箱即用的自适方案。按照 Bitpay 的建议，我们会用最近z个区块的中值大小y的一个倍数x来增大区块大小[12]。使用中值可以防止挖矿者用空或超大的区块去改变平均区块大小来作弊。x和z是网络必须遵从的硬性规定。

因此，最大区块大小是：

$$b = xy$$

8.4 货币单位与面额

QRL 的基本货币单位是叫 quantum (复数 quanta) 的代币。每个 quantum 再分为以下的较小单位：

- 1 : Shor
- 10^3 : Nakamoto
- 10^6 : Buterin
- 10^{10} : Merkle
- 10^{13} : Lamport
- 10^{16} : Quantum

因此，交易如果涉及一个 quantum 的一部分，就等于涉及非常多个 Shor 单位。

交易费以 Shor 单位支付和计算。

8.5 账户

用户结余储存在账户里。每个账户是个独一及可重复使用的账本地址，以用大写字母 Q 开头的字符串标志。

要建立地址，我们计算最高 XMSS 认证树的 merkle 根的 SHA-256 值，在在后面连接上一个 4 字节长的校验值（这个校验值是 merkle 根的双重 SHA-256 散列的头四个字节），最后再在前面加一个大写字母 Q。以 python 以伪代码显示就是：

$$Q \ + \ sha256(merkle根) \ + \ sha256(sha256(merkle根))[:4]$$

这是一个典型的账户地址：

$$Qcea29b1402248d53469e352de662923986f3a94cf0f51522bedd08fb5e64948af479$$

账户的结余以 quanta 作为单位，最小可以分割到一个 Shor 单位。

地址是状态的，每个交易使用新的 OTS 密钥对，QRL 储存每个账户所有以前使用过的公钥（这可以部分删减，因为可以随时重新计算出来，但需要用很多资源）。每当一个地址发出一个交易，我们就会把一个称为 nonce 的交易计数值增加。这使得没有储存整个区块链的钱包也能记得它们在状态 merkle 超树签名方法里的位置。

8.6 硬币发行

8.6.1 历史因素考虑

比特币是首个去中心化的加密货币，初时是实验性的，并且没有货币价值，因此用挖矿的形式来分发货币是合适的。最近 Zcash 采用了相同的方法，并把发行初期的一部分 coinbase 挖矿奖励的给予开源项目——而导致惊人的价格波动。

其他像以太坊的账本则以首次公开售币（ICO）的形式卖出一大部分初始硬币。这种做法的好处是早期采用者仍然可以通过支持项目而得益，同时项目本身也可以筹集资金来持续开发及成长。通过这个 ICO 方法，市场的发展也比较容易，因为投资者可以买卖更多的创世块硬币。

Auroracoin (2014) 则用另一个方法。它分配给冰岛每一个人相同份额的 ICO，而开发者则自己保留全部硬币供应的 50%。

其他密码货币不是完全复制比特币，就是用另一个基准代码来建立一条新链。

8.6.2 链与链之间转账

我们可以在 QRL 创世区块里嵌进当前比特币账本的状态快照，并基于这个快照来发行 QRL。一般概念是容许用户建立一个一次性"输入"交易，这个交易 含有一个独特的信息和签名（即是随机生成的 QRL 钱包地址，这个地址是用一个在快照一刻拥有比特币结余的地址的比特币私钥签名的）。这个特性可以保持到某个区块高度，之后余下的硬币供应就会以正常的方法挖掘。创世块的初始膨胀也会在这个区块高度被删减。这个方法的一个缺点是：虽然公平，它惩罚了拥有其他密码货币(除了比特币以外)的人，并且对新用户可能是技术上困难的。一个技术问题可能是：只需要从一个签名和消息就可以得到 ECDSA 公钥。这会把在程序中使用的比特币地址的公钥永久暴露，所以之后要把款项转移到新随机生成的比特币地址来缓解这个问题。

（？也可以给予以太坊拥有者同样的特性）

8.6.3 建议发行方法——草稿

QRL 首次发行将会是：

- ICO：在推出前，一百万个 quanta（最终硬币供应的 4.7%）。

• 所有构成初始 QRL 区块并大于 0.01btc 的比特币地址结余的快照。在区块高度达到 518400（3–6 个月）之前，任何人可以用节点钱包以 1：1 比例把硬币从比特币账本直接转移到 QRL 账本。

• 另外有一百万个 quanta 会被储存在一个创世块地址里给予基金会使用

• 余下供应可供挖 (21,000,000-(2,000,000+区块高度 518400 输入的比特币结余))

8.7 硬币发行时间表

比特币的一个特征是稀缺和相关代币的固定发行上限。在这方面，QRL 会跟随比特币，定下硬币供应的固定上限为21×10^6个 quanta。我们倾向于区块奖励以平滑指数衰减，直到硬币供应的硬性上限为止，这样就可以避免与比特币"减半"事件相关的波动。

硬币的总供应$x = 21\times10^6$，减去创世块生成的硬币y，会按指数式从Z_0永远向下减少。我们计算衰减曲线会持续 200 年分发挖矿奖励（到公元 2217 年，420480000 个区块，区块时间为 15 秒），直至只剩下一个 quantum 尚未被挖掘（虽然之后挖矿可以继续进行）。

在区块t，余下的硬币供应Z_t是：

$$Z_t = Z_0 e^{-\lambda t}$$

系数λ的公式是$\lambda = \frac{lnZ_0}{t}$，其中$t$是发行时间表里到最后一个 quantum 之前的区块总数。在区块 518400 之前，$\lambda = 3.98590885111 \times 10^{-08}$。每个区块的区块奖励$b$的公式是：

$$b = Z_{t-1} - Z_t$$

在创世块与区块 518400 之间，可以用输入交易来把比特币结余转移到账本。在区块 518401，发行时间表会重定目标去锁定新的输入结余、降低Z_t，及相应调校区块奖励。

参考文献

[1] http://oxt.me/charts.

[2] D. Bernstein. Sphincs: practical stateless hash-based signatures. 2015.

[3] J Buchmann. On the security of the winternitz one-time signature scheme.

[4] J. Buchmann. Xmss – a practical forward secure signature scheme based on minimal security assumptions. 2011.

[5] V Buterin. Ethereum whitepaper. 2013.

[6] A. Hulsing. W-ots+ - shorter signatures for hash-based signature schemes. 2013.

[7] A. Hulsing. Xmss: Extended hash-based signatures. 2015.

[8] A Karina. An efficient software implementation of the hash-based signature scheme mss and its variants. 2015.

[9] A. Lenstra. Selecting cryptographic key sizes. 2001.

[10] R. Merkle. A certified digital signature. CRYPTO, 435, 1989.

[11] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[12] S Pair. A simple, adaptive blocksize limit. 2016.

[13] Yonatan Sompolinsky. Accelerating bitcoin's transaction processing fast money grows on trees, not chains. 2014.

[14] A. Toshi. The birthday paradox. 2013.

增编

硬币发行更新：

以下是在翻译白皮书时提供而不包含在原文里的更新信息。信息更新了原文里的预售发行数字，取代了第 8.6 节：硬币发行和第 8.7 节：硬币发行时间表。这信息源自 Peter Waterland 的博文 https://medium.com/the-quantum-resistant-ledger/seed-investment-strategy-pos-algorithm-updates-3b3854e83a4a

硬币发行

- 最初分配为 6500 万个硬币（quanta）。

- 200 年后的最终分配为 1.05 亿个硬币（平滑指数衰减到硬性上限）。

- 团队将会持有预售的 20%（1300 万 quanta），其中的 65%将会归属于 QRL 基金会。

- QRL 基金会将会持有预售资金的 100%，其分期归属方案是与发展里程碑和研究目标相连的。

- 预售资金存款接受（比特币和以太坊多重签名合成地址（2-of-3、m-of-n 多重签名，密钥由三个不同的 QRL 创始成员分开持有）。

- 建议主网络在 2017 年第四季推出（可望到时在交易所同步上市）。