量子计算会是区块链的末日吗？量子计算领域的最新进展让很多人对经典密码学的前景感到担忧。例如，谷歌开发了一种具有72个量子位的量子计算芯片，这与破解经典密码学所需的数千个可用量子位相差甚远。似乎不再那么遥不可及。有人估计，到2031年，RSA和ECC(椭圆加密)算法被破解的概率约为50%，而比特币和以太坊等区块链使用的是经典密码学，虽然比特币使用的是双SHA256算法。相比银行和支付宝使用的安全系统，它还是有很多攻击媒介的。量子计算似乎正在成为区块链的达摩克利斯之剑，我们需要担心吗？译者认为我们需要密切关注，但我们没有&#039；不需要太担心。在真正紧急的情况下，社区可以通过硬分叉的方式替换比特币等区块链使用的加密算法，区块链业界也有研究人员在不断研究相关的反量子加密算法，来自联盟链团队R3的研究团队。在分析了相关攻击向量后，提出了一种反量子签名方案。从结果来看，这种方案更适合R3的Corda和SuperBookFabric。

以下是论文翻译：摘要——受区块链体系结构和现有的基于Merkle树的签名方案的启发，我们提出了BPQS，一种可扩展的抗后量子(PQ)的数字签名方案。它最适合区块链和分布式账本技术(DLT)。该协议的一个独特之处在于，它可以使用特殊的链或图像结构来降低密钥生成、签名和验证的成本以及签名的大小。与最近的其他改进相比当一个密钥被合理数量的签名重用时，BPQS将优于现有的哈希算法，如果需要，它还支持允许无限数量签名的回滚机制。我们提供了该协议的一个开源的具体实现方案。并进行了基准测试。与本文相关的术语：后量子加密、数字签名、分布式账本、区块链、Merkle树I引言量子计算领域的最新进展及其对经典密码学的威胁。它刺激了更多的人&#039；美国对后量子(PQ)研究的兴趣。更具体地说，由于Shor算法[1]，量子计算机可以很容易地在多项式时间内分解大整数因子，从而有效地破解RSA。Shor算法的应用可以解决离散对数问题，而今天&#039；的数字签名(如DSA、ECDSA和EdDSA)使其无效[2]。建造量子计算机的竞赛已经开始。像谷歌、微软、IBM、D-Wave、英特尔这样的公司已经处于领先地位。然而到目前为止，我们还无法建造一台拥有数千个稳定量子位的计算机，可以让经典的公钥密码技术过时。然而，在这个领域已经取得了显著的进展，一些乐观主义者预测，在未来的10到20年[3]和[4]大型量子计算机或许能够破解不对称加密。破解公钥加密系统会带来很大的安全影响。几乎所有的东西都来自HTTPS、VPN和PKI，其中涉及到RSA或椭圆曲线加密算法(ECC)的安全性。。区块链领域也将受到打击，它可能是受影响最大的领域之一，因为黑客可以从中获得经济激励，他们可以匿名访问区块链账户。为了解决密钥泄露的问题后量子密码术已经开始兴起，它将保护我们免受量子霸权的冲击。我们提出的BPQS解决方案是基于XMSS方案的修改版本[5]。它实际上使用单一的认证路径；因此，是链条，不是树。它只需要关注{一次性且更少}的密钥，也就是说，它通常最适合区块链(因为我们希望保持匿名，尽量减少跟踪)。与现有方案相比，我们的方案在只需要签名一次或几次的情况下会有更好的性能。。与一次性签名方案(OTS)不同，BPQS方案提供了回滚机制，可以方便地支持多重签名。据我们所知，这是第一个可以利用现有区块链或图形结构来降低签名成本的签名方案(即使我们计划多次签名)。。这使得现有的多态签名方案对

于区块链应用来说已经过时。此外，事实上，BPQS完全基于哈希函数，因此它的实现不需要特殊的数学理论，这使得它成为现有或新区块链应用程序的强大签名方案候选。。其次，量子计算是由市场对更高效计算的需求和解决以前不切实际的问题的能力驱动的。量子计算从基础理论研究走向实用研究的进程正在快速推进。十年前，几乎没有实用量子计算机的证据。但到2018年，，谷歌推出了72量子位的新型量子计算芯片Bristlecone[6]，比IBM2017年宣布的50量子位处理器提前了22量子位。我们还应该提到D-Wave公司。，该公司最近发布了一款2000量子位处理器[7]。但据报道，D波的量子加速分析值得商榷[8]。总之，虽然我们还需要更多的研究和实验，但我们可以保持量子芯片的稳定和低错误率。但是，量子计算的技术在进步是不争的事实。a.对密码学的影响量子技术带来新的安全挑战。如上所述，它将削弱经典密码学的前景。由于Shor算法，广泛使用的基于离散对数的公钥密码系统认为会在后量子(PQ)时代破解。根据一些估算[3]，到2031年，RSA和ECC算法被破解的概率约为50%。此外，Grover算法[9]也可能影响对称加密和哈希。但目前，我们不&#039；t不知道如何获得比传统计算机更多的二次加速，所以我们可以通过增加key/output的大小来保证PQ的安全性。我们还应该关注量子技术的发展，一些怀疑论者[10]认为。量子霸权永远不会达到几千个可用量子比特的水平[2](也就是破解经典密码学所需的水平)。虽然量子计算机什么时候能达到这个水平还存在不确定性，但是在学术界和工业界的研发还是有意义的。。一些人也试图结合经典和后量子算法[11]和[12]，以更好地为量子天启做准备(假设它会发生)。这里还应该提到标准化组织，如NIST它已经开始研究标准化后量子(PQ)算法[13]，[14]。欧洲电信标准协会(ETSI)更加谨慎，建议任何需要将加密数据存档到2025年(或更长时间)的组织。，应该是担心量子计算机的威胁[15]。区块链和分布式账本领域的人应该更加关注量子计算，因为公钥持有的资产/硬币可能需要几十年的时间。。b.对区块链和分布式账本技术的影响(DLT)比特币、以太坊等传统区块链使用经典的公钥加密技术对交易进行签名，这些网络被认为容易受到量子计算的攻击。其他系统如zCash和Quorum。，严重依赖于特殊的椭圆曲线来提供零知识证明功能，而一个ECC漏洞将威胁到这些书的完整性[16]。这将是一个重大的安全隐患，导致网络被彻底破解。大多数区块链使用公钥加密哈希生成的地址来减轻这种威胁。这种额外的安全层意味着公钥只有在参与第一笔交易(即消费比特币)后才会暴露在账本中。。直到此时，才暴露出哈希接收方密钥(地址)，所以像Shor算法这样的攻击不适合这个阶段。然而，即使使用了散列密钥，以下攻击媒介仍然适用：1.地址重用：当一个事务被签名时，公钥将被泄露。因此，与其关联的地址不再安全。虽然我们可以建议每笔交易使用一个新的地址/密钥，但是旧的比特币客户端和一些矿池仍然会重用地址；2.废弃的硬币/资产：如果它们的相关地址不是通过散列法生成的。这些旧地址的公钥会暴露，比如2012年之前的比特币；3.正在进行的交易：一旦您在网络上广播了一项交易，但该交易未被区块链接受，这些交易就容易受到攻击。当然这种攻击的窗口期机会有限，但理论上还是有可能的。在合法执行交易之前，攻击者可以恢复私钥，然后用它签署另一个交易，并将资产转移到自己的地址。4.事务拒绝/失败：例如，如果签名的事务失败密钥会因为给的交易费太低，或者恶意

方阻止交易中继，或者验证过程中出现脚本错误而被攻击；5.多重签名事务/混合事务：如果使用CoinJoin协议[17]这将在交易完成之前向其他方透露公钥；6.公布单一地址：公布和使用同一个地址，比如集资，会暴露第一笔消费交易的公钥，所以会使后续的资金收益面临风险。社区通常建议的解决方案是升级签名算法。这样它就能抵抗量子计算。然而，这将总是导致区块链中的硬分叉，这将引入各种复杂性。也有一些区块链解决方案已经支持后量子技术。例如，QRL[18]，IOTA[20]和Corda[22]，以及BPQS签名方案的应用可以减少签名大小，并提供回滚机制以允许具有相同密钥的多个签名。。第三，使用同一个密钥对OTS哈希方案进行签名的场景要求密钥只能使用一次，否则安全性将被破坏。然而，在区块链中，有许多情况需要对多个签名使用同一个密钥：1.如前一节所述，交易可能会失败或被拒绝。，这需要额外的签名来解决。2.密钥所有权的证明，其中甲方需要在(挑战)乙方提供的消息上签名，然后让乙方验证这个所有权；3.偿付能力证明、最低资产要求的所有权证明。没有透露任何进一步的信息。解决这个问题的方法是进行独立审计，并将其记录在&quot发送给自己&quot与业主的交易&#039；的私钥；4.一种分叉的区块链，其中地址(和相关的关键字)被复制到分叉的链上。，然后在每个分叉链中使用，来自同一个地址的重复交易(OTS签名方案)违反了一个密钥只能使用一次的条件，因此OTS签名方案的强度将减半。5.资产的战略性双重支出，可导致交易拒绝。在这种情况下，也可能是一个故意的锁使得一个等待的事务被拒绝。这种情况可能发生在意外支出的场景中。用同一个密钥签署一个重复的事务会导致两个事务同时失败，而且会导致不好的后果，就是密钥被重用。四、基于hash的后量子(PQ)数字签名a.一次性签名(OTS)基于hash的签名方案可以追溯到1979年，这得益于LamportOTS(一次性签名)方案[24]。。Lamport方案背后的逻辑很清楚：签名者生成一对每个比特都需要签名的随机值，并形成一对私钥。公钥由这些值的散列组成。为了签署消息，签署者逐位读取消息并显示一个值。每个秘密对取决于比特值。然后，验证者可以验证所有秘密值的散列是否等于公钥中的相应散列。虽然LamportOTS哈希计算被认为是快速的，但它的密钥和签名大小相对较大。例如如果使用SHA256作为底层哈希函数，则公钥由512个256位哈希输出组成(每位1个哈希对)，签名由256个秘密值组成(每个256位)。如果我们总结密钥和签名数据，它们需要占用24.5kB，同样，如果使用SHA512算法，大约会占用98kb；对原始算法[19]、[25]和[26]的进一步增强可以显著减小密钥大小。现在WOTS算法[19]及其变体被认为是最有效的密钥和签名压缩方法，而Bleichenbacher和Maurer[26]的图形方案试图在每个消息位的签名大小和哈希函数求值次数方面达到最佳效率。值得注意的是，OTS方法之间的主要区别之一是所需的安全假设是否能抵抗反哈希函数，以及是否使用了额外的位掩码。目前，WOTS-T[27]在QROM模型中被证明是安全的。，它被认为是WOTS级数方案[19]中最有希望的候选方案，因为只需要一个额外的种子值与公钥一起计算所需的位掩码，其安全性不会受到&quot生日悖论&quot。它还介绍了所有散列的关键函数调用。为了防止多个目标进行第二次原图攻击。后者导致更短的公钥和散列输出大小。b.少签名和多重签名虽然将一次性签名方案转化为多重签名方案的方法有很多[5]，[28]:[30]，但它是一种比较流行的方法。，就是

使用Merkle认证树。使用Merkel树，可以发布的签名总数在密钥生成时定义。这种方法的主要优点是它的短签名输出和快速验证。但是，缺点是密钥生成时间相对较长，而且它们是有状态的。图1描绘了4度(最大)默克尔树签名方案。

图1:可以签署多达4条消息的默克尔树签名方案。如果我们用OTS1方案签名，深色节点表示所需的认证路径。然而，切换到无状态的少签名方案需要额外的复杂性和更大的签名输出。。HORS[29](及其扩展HORST[23])方案是最常用的多重无状态签名方案，如SPHINCS[23]。多重散列签名方案可以通过组合上述{一次及更少}方案来构造。并且可以分为两类：有状态的(如XMSS，LMS)[31]和无状态的(如SPHINCS，SPHINCS，Gravity，Simpira，Haraka)[23]，[32]:[35]。。有状态方案通常产生短签名，但是它们需要一种机制来维护状态(使用了哪些路径/密钥)。另一方面，无状态方案在顶部以中等大小的默克尔树或树层开始，而不是在底部使用OTS签名。他们使用不太常见的签名方案。后者允许它们随机选择索引，因此不需要跟踪路径状态。无状态方案的缺点是它们的签名大小，例如，在SPHINCS-256[23]方案中。每个签名的大小将是41kB。次要散列签名方案和多重散列签名方案之间并不总是很清楚。在文献中，次要签名方案通常指无状态方案，如芭比[28]，霍斯[29]和霍斯特[23]。但是在很多实际应用中，这些签名仍然是不够的。另一方面，多重签名方案可以被配置为允许在高度交互的环境中重用相同的密钥对(可以使用很多年)。。GravitySPHINCS[33]方案的作者声称1万亿(2^40)个签名是一个合理的上限，而SPHINCS-256[23]签名方案最多允许1万亿(2^50)个签名。实际上，人们可以将多重签名方案参数化，以支持几个或多个签名。c.散列函数的速度和安全性对于所提议项目的整体安全性显然是非常重要的。几个因素会影响算法的选择。，包括速度、安全级别和可用性；例如，哪些硬件功能可以用来提高运行时性能？然而，首先要确定的是，该算法应该能够抵抗后量子(PQ)攻击。SHA-2和SHA-3算法支持多种摘要大小。，即224。256.384和512位[36]、[37]。我们通过提高Grover算法提供的搜索速度来观察这一点，碰撞阻力可以从所选摘要的一半大小减少到三分之一。所以在大规模量子计算机的情况下。384位版本的SHA-2和SHA-3算法将提供128位防冲突安全性。256位版本只能提供85位安全性[4]。此外，我们已经观察到量子图像攻击是针对256位版本的SHA-2和SHA-3算法。，分别由2个153.8和2个146.5代码循环完成[38]。通过这两点观察，基于hash安全性的考虑，SHA256算法其实并不合适，但还是比较安全的。还应该提到的是与经典的vanOorschot和Wiener哈希碰撞算法相比，即使在量子计算机发展速度的乐观假设下，后量子(PQ)算法的性价比也会更差[39]。根据ebac[40]中提供的性能测量我们在通用CPU上评估了SHA-2、SHA-3和BLAKE2算法的相对性能。我们特别选择了英特尔、AMD和ARM处理器，以覆盖典型的台式机和笔记本电脑。从表1中我们可以看出每个字节的周期数随着输入的大小而减少，当输入超过块大小时，下降率自然会趋于平缓。还应注意的是，不同版本的SHA-3通常比它们的SHA-2对手表现更差。原因之一。原因是SHA-1和SHA-2拥有来自现代处理器的更多硬件支持(如英特尔SHA扩展提供的指令集扩展)。请注意，尽管SHA-2算法不提供针对长扩展攻击的保护，但它提供

了类似于SHA-3的位安全性。。一般来说，基于hash的后量子签名方案(包括BPQS)不容易受到这种攻击。因此，我们将考虑使用SHA-2来比较SHA-2和SHA-3，因为它具有更好的性能优势。如果性能很重要也可以考虑采用支持度较小的BLAKE2b[41]算法。但要强调的是，这种算法与上述算法相比，会缺乏广泛的库支持。表1:SHA-2、SHA-3和布莱克的性能指标

a:根据ECRYPTII[40]在eBACS中报告的数字；英特尔：amd64。源氏122。Supercop-20171020:AMD:amd64。源氏262。Supercop-20171020:ARM:Armeabi，Odroid，Supercop-20160806V.大部分(如果不是全部的话)为一次性密钥定制的区块链量子签名。。基本Merkle树签名方案可以签名的最大信息量是2h，其中h是树的高度。此外，在密钥生成期间，应计算所有子叶(密钥)以形成根。出于上述原因要构建一棵高度为h=40的树，这样的密钥生成会被认为是不切实际的，因为我们需要计算2^40个OTS密钥，每个OTS密钥需要多次哈希调用(例如，兰波特OTS需要512次哈希调用)。当使用SHA256的WOTS(w=16)时，需要67哈希调用。在保持密钥生成时间可行的同时，还可以允许大量签名使用多级树。BPQS是XMSS类协议的简化单链变体[5]。从字面上看，它是基本的默克尔树签名方案的扩展(见图1)。理论上，BPQS可以多次签名，但其设计侧重于短而快的一次性签名，必要时还有重新签名的附加选项。。上述要求是典型的区块链或分布式账本技术(DLT)所要求的，因为匿名可以通过使用一次性密钥来维护。然而，许多事情可能会出错。例如，一个事务可能没有通过，或者在区块链中可能有一个分叉。在这种情况下，，应该能够多次签名，而不会危及安全或冻结资产(见IOTA&#039问题[21]和[42])。此外，BPQS的另一个优点是，它也是相反需求(用同一个密钥多次签名)的理想候选。这个有趣的属性是，与其他基于哈希的后量子(PQ)解决方案相比，区块链和分布式账本系统的底层图形结构有效地允许以最小的成本进行多重签名。。这是通过引用过去使用过的相同BPQS键的块(或事务)来实现的。简而言之，只需要提交少量的新签名。事实上，因为之前的交易已经在账簿上验证过了，所以验证者不需要验证路径的其余部分。因为过去已经验证过了。这个特性使得BPQS对于基于公证的分布式书籍特别有用，比如Corda[22]和Fabric[43]，因为公证节点通常用相同的已知密钥签署事务。。BPQS计划BPQS需要一个基层ots计划。理论上，任何OTS方案都可以应用，但我们的方案的逻辑与XMSS系列协议的逻辑相同，所以我们将选择WOTS[19]变体。L树和位掩码的生成用于定义安全假设和证明，类似于XMSS[5]及其多级版本XMSS-MT[44]和XMSS-T[27]。再者，按照[39]的说法，使用量子算法的抗碰撞性其实更便宜。因此，它类似于重力SPHINCS[33]方案，并且可以省略位掩码和L树。你也可以说BPQS是XMSS系列方案的子集，主要针对快速一次性签名。主要区别在于XMSS通过使用哈希树克服了每个密钥对消息的限制，这将把许多OTS验证密钥的真实性降低到一个普通的XMSS根密钥；取而代之的是，BPQS使用包含一个小的2子叶默克尔树的链。从几何定义来看，XMSS将在宽度和高度上都增长(见图1)，而BPQS将只在链高度上增长(见图2)。总之，我们强调BPQS是一个普遍的区块链结构，其中的块是&quottiny&quot默克尔树，这意味着根据应用程序

的要求，可以参数化。如果应用掩码，其确定性生成应遵循与相应XMSS系列方案相同的逻辑。BPQS签名方案有两个基本构件：BPQS-少数，它严格支持次要签名。如图2(左)所示；BPQS-EXT:理论上可以扩展到支持多个签名，如图2(右)；在BPQS-worth中，所有的密钥都是在密钥生成过程中预先计算好的，每个额外签名的代价只是一个额外的哈希输出。但不能扩展到实际支持&quot无限制&quot签名。另一方面，BPQS-EXT最初只需要两个OTS键，这与BPQS-少数相反。在OTS回滚键的每棵二子叶默克尔树的左叶中，必要时，其可用于签署下一个块。不幸的是，可扩展属性是有代价的，每个新签名都需要一个额外的WOTS键。

图二：BPQS-少(左)，一个次要的签名方案。BPQS-EXT(右侧)线性可伸缩多重签名方案；完整的BPQS模式在某种程度上结合了BPQS-少数和BPQS-EXT(其中BPQS-少数链中的最后一个子叶在BPQS-EXT回滚键中)。这项技能，它允许我们将少签名方案的变体转换为多签名方案。事实上，BPQS-
EXT可以被认为是BPQS的一个特例，其中它没有初始的BPQS-少数链。

图3:完整的BPQS协议，它有一个高度为3的BPQS-少数顶层和一个BPQS-EXT回滚键以允许可扩展性。BPQS的参数包括：1.使用的WOTS变体(例如，WOTS-T[27])；2.Winternitz参数(例如，w=16)，它定义了解释初始哈希的基础。与XMSS[5]类似，w定义了每个WOTS链的实际大小，这反过来会影响签名的大小。请注意文献中对温特尼茨参数的解释没有一致意见。例如，在LMS[45]中，定义为2w，所以wBPQS=16=2^4，相当于wLMS=4.3和底层哈希函数(如sha384)；4.预先计算的OTS键的数量，即初始值高度(例如，h=4)；b.混合BPQSBPQS的扩展性属性可以实现各种自定义结构。BPQS可用作构建块，将任何散列签名方案转换成{一次性或更少}优化方案。例如，在图4中，BPQS-少数用作第一个(较短的)签名，然后它回滚到另一个后量子(PQ)方案。尽管在所描述的方法中，回滚(其他)之后的量子(PQ)方案的密钥对应该是先验已知的并且是预先计算的。您也可以以类似的方式使用BPQS-EXT，所以这不一定是必需的，并且&quot其他&quot后量子(PQ)密钥将仅在(几个)签名用尽时生成。此外，如果&quot其他&quot后量子(PQ)方案是无状态的。例如SPHINCS，那么最终的协议几乎是一个&quot启动时有状态，然后无状态&quot。应该强调的是&quot其他&quot后量子(PQ)方案可能是另一个BPQS方案，因此最终可以创建一系列不同的BPQS方案。。后者会导致签名更短，一次只使用BPQS-EXT来扩展。至于&quot其他PQ关键参数&quot如图4所示，需要一些方案发布位掩码(或XMSS-T[27]中的种子)作为初始公告公钥的一部分，这一点很重要。。否则，它将允许对手在伪造活动中选择种子/位掩码。然而，如果BPQS使用具有更大输出的散列函数(例如SHA384或SHA512)，这可能是不必要的，因为它提供了对抗量子碰撞攻击的安全性。

，仍然足以阻止这样的攻击。图4:一个通用的BPQS协议(BPQS-VERS1),顶层BPQS-少，高度为3。，其最后一个根是另一个后量子(PQ)方案的公钥，如XMSSMT[44]

或SPHINCS[32]。c.如有必要，组合量子(PQ)方案如上所述。bpq可以回滚到另一个后量程(PQ)方案。通过应用类似的逻辑，图5显示了将多个后量子(PQ)方案组合成一个方案的各种定制模型。这个方法非常简单，但是它允许一个非常有用的结构。例如，图5A和5b中的有状态和无状态方案，或者图5C中的具有无状态回滚的有状态方案。后者为集群环境(其中多个节点需要就签名状态达成共识)和回滚机制提供了解决方案，是在共识因某种原因失效时，保证系统能够继续运行(维持其功能)的前提条件。

图5:使用并行BPQS逻辑将多个方案组合成实际后量子(PQ)方案的各种推荐设计。注意，如果底层方案需要额外的参数(比如位掩码)，这些参数应该与根公钥一起发布，类似于图4中所示的三个描述的方案。就以下两个因素而言，它提供了不同程度的灵活性：选择密钥生成时间和签名大小之间的平衡；稍后，决定是否允许选择底层算法；例如，选项B需要生成两个PQ密钥，以形成要公布的组合公钥。同时，选项A实际上是一个BPQS-EXT，它将用于签署&quot即将到来的&quotPQ方案。沿着相同的路线，选项C结合了A和B，但是左边的PQ方案不需要通过先验算法进行选择和计算。注意我们甚至可以组合两种不同的有状态或无状态模式。例如，如果为了兼容，在两个不同的区块链中使用相同的密钥，则其中一个支持原始的SPHINCS-256[23]方案。而另一个支持它的变体(或它的标准化版本)。不及物动词实验结果本节将展示基于大量实验的BPQS方案的性能分析，并将其与一系列最先进的签名方案进行比较。选择这种对比方法因为它在今天很流行&#039；公钥基础设施和区块链社区。我们的实验结果是基于BPQS签名方案的原型实现[46]，其中包括如何再现基准测试的细节。我们选择的系统规格。，包括八核&#039；s英特尔酷睿i7-7700HQ处理器(2.80GHz)和15.5GB内存，而系统运行在Linux4.13.0-38和JRE1.8.0_161。关于实现，，标准JCE已用于哈希函数调用，而它用于其他方案的实现，这些方案分别基于BouncyCastle(ECDSA，RSA，SPhincs-256)[47]和i2p(EdDSA)[48]库。。性能比较表2比较了各种签名方案的性能，包括使用SHA256和SHA384作为底层哈希函数的BPQS签名方案(w=4和w=16)。报告结果的平均运行时间，以毫秒为单位。。我们选择消息列表而不是单个消息或少量信息的主要原因是为了避免签名和验证操作中可能出现的任何偏差。表2:密钥对生成时间(毫秒)，签名和验证第一条消息的时间(毫秒)。

需要强调的是，我们目前实现的BPQS方案并没有针对并行处理进行优化，它不会缓存WOTS哈希迭代的中间结果。人们已经认识到，缓存密钥秘密可以大大加快签名的处理速度[33]。在实践过程中因为BPQS签名方案最适用于只签名一次或多次的应用程序，所以OTS密钥的路径和数量相对较小，易于放入内存。如果在密钥生成期间所有完整的WOTS结构都存储在内存中，那么签名只不过是一些内存查找。这不&#039；不涉及运行任何散列操作，这不包括前面要求的消息散列。即使没有并行化或缓存，BPQS签名方案的性能也接近经典和后量子(PQ)数字签名算法的性能。应该强调的是BPQS的最简单形式(BPQS-EXT)已经在上面的比较中使用了，其

中生成了两个WOTS系列密钥，其中一个用于签署第一个消息，另一个负责签署回滚操作。我们希望区块链键的平均使用次数是1次。但是附加签名只在极少数特殊情况下进行，所以我们的比较也集中在第一次签名操作上。在实践中，我们希望区块链钱包可以被缓存和并行化，以进一步提高性能。。就签名大小而言，所有BPQS模式在前(log2(h)1)次的性能都优于XMSS签名方案。BPQS签名的大小随着密钥被重用的次数线性增加，因此签名输出的长度是动态的。一开始会很小。，但会随着签名数量的增加而增加。参数应该是初始密钥生成成本和签名大小之间的折衷。在最好的情况下，签名的大小将随着每个附加签名的散列输出大小的增加而增加，而在最坏的情况下，

，它随着WOTS的大小而变化[19]。图6:不同模式的BPQS和XMSS[5]签名方案支持32个OTS密钥的签名大小比较。所有方案的签名输出都是|WOTS|x|H|，其中x是完全签名路径所需的附加哈希数。在这张图表中，，x轴代表签名数量，y轴代表x；在大多数BPQS模式下，第一个签名输出的大小是|WOTS||H|，而对于XMSS，它的签名输出大小是|WOTS|h|H|。，其中h是所用的默克尔树的高度。当一个最常用的WOTS(w=16)被用作底层OTS方案时，每个WOTS的大小为|WOTS|，等于67|H|。，其中|H|是基础哈希函数的摘要大小(例如，对于SHA256，它等于256位)。图6描述了以下方案的每个附加签名的签名大小：BPQS-少数(图2的左侧)，其中h=32XMSS[5]，h=5。其中h代表树的高度；BPQS-VERS1(图4)，回滚到XMSS，高度h为5；BPQS-GEN-B(图5B)，右边是BPQS-少数，左边是XMSS。，两人身高都是5；七。结论和未来工作在这篇论文中，我们介绍了BPQS签名方案及其扩展方案，以支持{一次较少}优化的量子(PQ)签名方案。我们还提出了区块链和分布式账本技术即将面临的安全挑战。然后我们解释为什么我们不&#039；不推荐纯OTS方案作为反量子计算的替代品。如上所述，BPQS比传统的非量子签名方案(如RSA、ECDSA和EdDSA)有一些优势，可以提供更可靠的量子安全性。这是因为它的加密哈希函数会更安全。其中，BPQS协议的主要特点包括：1.更短的签名和更快的密钥生成。当签名只做一次或几次时，其签名和验证时间将比XMSS[5]和SPHINCS[23]系列的后量子(PQ)协议更短，并且在区块链系统中，还可以实现更好的匿名性；2.在计算上相当于非量子方案。。人们可以利用易于使用的多哈WOTS进行并行化和缓存，从而提供几乎即时的签名和更快的验证。3.它的可扩展性允许多个签名，如果需要，也可以很容易地定制。它还可以回滚到另一个多重签名方案；4.当它应用于区块链和分布式分类帐应用程序时，它可以通过引用以前的事务(其中相同的键将被重用)来利用底层的链或图结构。这可能意味着每个新的BPQS签名只需要一个OTS方案的努力，因为剩余的到根的签名路径已经在帐簿中，所以它们可以被省略；5.它可以作为构建块来实现新颖的PQ方案，例如同步有状态和无状态方案，这可能有利于集群环境。当共识消失时，其中的节点可以回滚到无状态方案；此外，这样的方案可以用于向前和向后兼容的目的，或者当需要在两个独立且不兼容的区块链中重用密钥时；这种原始BPQS协议的主要缺点是其签名输出的大小将随着签名的数量线性增加。但是，我们可以使用组合PQ方案或使用区块链应用程序中现有的图形

结构来减轻这种影响。总之，BPQS具有可定制性、缓存性和可扩展性。它可以作为无状态、有状态和其他PQ方案之间的桥梁协议。