B 编码验证

在t时刻，K个区块X1(t);X2(t).......XK(t)被提交到K碎片进行验证。PolyShard方案通过三个步骤验证这些区块。

第一步：区块编码。从接收到的K个区块，每个节点i计算编码块~Xi（t）作为线性组合

使用与（9）中相同的系数集。就是。我们注意到这种编码操作也可以

在点被视为评估多项式，该步骤引起O（NK）操作

通过网络，因为N个节点每一个都计算K个区块的线性组合。

第二步：局部计算。每个节点i将函数应用于编码块，以及本地存储的编码子链来计算。此步骤需要在整个网络中进行总共Nc（ft）操作

验证操作需要c（ft）步。完成本地计算后，每个节点i向其他所有节点广播其计算结果。

第三步：解码。使用计算结果，其中为恶意节点的最大值是函数，每个节点解码预期结果。因为是一个度为（K-1）d的单变量多项式，可以被看作在点的测评，并且它可以被遵循具有维度（K􀀀1）d + 1和长度N的Reed-Solomon码的解码过程之后被恢复（参见例如[38]）。

这里应该有图片。

为了让解码强大到uN恶意节点，我们必须有。换句话说，一个节点可以成功编码，当且仅当K碎片的数量上限。基于这个理论，我们将PolyShard算法中碎片的数量设置为，线性的符合网络大小N。

在每个节点解码长度为N的Reed-Solomon码的复杂度为O（N log2 N log logN），并且解码步骤总复杂度是O（N2 log2 N log logN）。通过编码，每个节点评估w1......wk来恢复，从而获得结果。

C 增量子链更新

验证过程失败的块（即ztk= 0）应该被最终链中的空块替换，所以我们有，Yk（t）= ztk

XK（t）。每个节点现在计算：，并且将Yi（t）附加到其本地编码子链中以将其更新为。更新子链与块编码步骤具有相同的计算复杂度，即O（NK）。备注5.由于（10）中的系数集合iks与（9）中的系数相同，因此将编码块附加到编码的子链中等效于将未编码的块附加到未编码的子链，然后从更新的子链进行编码。 这个子链增长和存储编码之间的可交换性允许每个节点递增地更新其本地子链通过仅访问新验证的块而不是整个块历史记录。

验证和存储更新过程中的操作总数为O(NK)+Nc（ft）+O(N2log2NloglogN)，其中O(NK）+O(N2log2NloglogN)为额外的编码开销比较使用未编码的分片方案。因为K<=N，编码开销减少到O(N2log2NloglogN）。Polyshard算法吞吐量为：



D PolyShard的性能分析

通过证明Polyshard对uN=0（N)的稳健性，我们分析存储效率和PolyShard的吞吐量。对于验证函数ft具有恒定度d的场景，计算复杂度跟链长t成正比。这适用于扫描整个区块链历史记录的验证过程。对于每一个以往的区块，执行具有固定复杂度的操作中的平衡检查。对于恒定维度d，我们有，存储效率为θ（N）。当c（ft）=O(t)，计算ft的复杂度随着t的增加而增加，（11）中的吞吐量变为

这完成了定理1的证明。

我们可以看到，由于PolyShard的编码和解码操作的复杂性不能随着t而扩展随着链的增长，编码开销变得无关紧要。 PolyShard方案同时实现最佳缩放安全性，存储效率和吞吐量。