黑客派

# 区块链学习系列——PFBT 研究

作者：chensy

原文链接：https://hacpai.com/article/1499929586765

来源网站：黑客派

许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

本文主要是从网上搜集整理的，主要是进行梳理，理解PFBT。
<h2 id="toc_h2_0">1.拜占庭将军问题是什么? </h2>
拜占庭将军问题是一个共识问题：首先由Leslie Lamport与另外两人在1982年提出，被称为Theyzantine Generals Problem或者Byzantine Failure。核心描述是军中可能有叛徒，却要保进攻一致，由此引申到计算领域，发展成了一种容错理论。随着比特币的出现和兴起，这个名问题又重入大众视野。
<h3 id="toc_h3_1">1.1. 拜占庭将军问题场景</h3>
关于拜占庭将军问题，一个简易的非正式描述如下：
拜占庭帝国想要进攻一个强大的敌人，为此派出了10支军队去包围这个敌人。这个敌人虽不比拜庭帝国，但也足以抵御5支常规拜占庭军队的同时袭击。基于一些原因，这10支军队不能集合在一起点突破，必须在分开的包围状态下同时攻击。他们任一支军队单独进攻都毫无胜算，除非有至少6支队同时袭击才能攻下敌国。他们分散在敌国的四周，依靠通信兵相互通信来协商进攻意向及进攻时间困扰这些将军的问题是，他们不确定他们中是否有叛徒，叛徒可能擅自变更进攻意向或者进攻时间。这种状态下，拜占庭将军们能否找到一种分布式的协议来让他们能够远程协商，从而赢取战斗？这就著名的拜占庭将军问题。
应该明确的是，拜占庭将军问题中并不去考虑通信兵是否会被截获或无法传达信息等问题，即消传递的信道绝无问。Lamport已经证明了在消息可能丢失的不可靠信道上试图通过消息传递的方式达一致性是不可能的。所以，在研究拜占庭将军问题的时候，我们已经假定了信道是没有问题的，并在个前提下，去做一致性和容错性相关研究。如果需要考虑信道是有问题的，这涉及到了另一个相关问：两军问题。
<h3 id="toc_h3_2">1.2.与拜占庭将军相关问题：两军问题 </h3>
正如前文所说，拜占庭将军问题和两军问题实质是不一样的。国内大量解释拜占庭将军问题的文将两者混为一谈，其实是混淆了两个问题的实质，由此造成了许多误解。这两个问题看起来的确有点似，但是问题的前提和研究方向都截然不同。

如图所示，白军驻扎在沟渠里，蓝军则分散在沟渠两边。白军比任何一支蓝军都更为强大，但是军若能同时合力进攻则能够打败白军。他们不能够远程的沟通，只能派遣通信兵穿过沟渠去通知对方军协商进攻时间。是否存在一个能使蓝军必胜的通信协议，这就是两军问题。
看到这里您可能发现两军问题和拜占庭将军问题有一定的相似性，但我们必须注意的是，通信兵经过敌人的沟渠，在这过程中他可能被捕，也就是说，两军问题中信道是不可靠的，并且其中没有叛之说，这就是两军问题和拜占庭将军问题的根本性不同。由此可见，大量混淆了拜占庭将军问题和两问题的文章并没有充分理解两者。
两军问题的根本问题在于信道的不可靠，反过来说，如果传递消息的信道是可靠的，两军问题可。然而，并不存在这样一种信道，所以两军问题在经典情境下是不可解的，为什么呢？
倘若1号蓝军（简称1）向2号蓝军（简称2）派出了通信兵，若1要知道2是否收到了自己的信息，必须要求2给自己传输一个回执，说“你的信息我已经收到了，我同意你提议的明天早上10点9分准时攻”。然而，就算2已经送出了这条信息，2也不能确定1就一定会在这个时间进攻，因为2发出的回执并不一定能够收到。所以，1必须再给2发出一个回执说“我收到了”，但是1也不会知道2是否收到了样一个回执，所以1还会期待一个2的回执。
虽然看似很可笑，但在这个系统中永远需要存在一个回执，这对于两方来说都并不一定能够达成足的确信。更要命的是，我们还没有考虑，通信兵的信息还有可能被篡改。由此可见，经典情形下两问题是不可解的，并不存在一个能使蓝军一定胜利的通信协议。
不幸的是，两军问题作为现代通信系统中必须解决的问题，我们尚不能将之完全解决，这意味着我传输信息时仍然可能出现丢失、监听或篡改的情况。但我们能不能通过一种相对可靠的方式来解决部分情形呢？这需要谈到TCP协议。事实上，搜索“两军问题与三次握手”，您一定可以找到大量与TP协议相关的内容。若您是通信方面的专家，权当笔者是班门弄斧，这里仅用最浅显易懂的方式科普TP协议的原理和局限，可能存在一些毛刺，请多包涵。

TCP协议中，A先向B发出一个随机数x，B收到x了以后，发给A另一个随机数y以及x+1作为答复这样A就知道B已经收到了，因为要破解随机数x可能性并不大；然后A再发回y+1给B，这样B就知道A经收到了。这样，A和B之间就建立一个可靠的连接，彼此相信对方已经收到并确认了信息。
而事实上，A并不会知道B是否收到了y+1；并且，由于信道的不可靠性，x或者y都是可能被截获

，这些问题说明了即使是三次握手，也并不能够彻底解决两军问题，只是在现实成本可控的件下，我们把TCP协议当作了两军问题的现实可解方法。
<img src="https://static.hacpai.com/images/img-loading.svg" alt="" data-src="http://www.btckan.com/public/resources/pic/news/2015/10/16/hup7920WU0.png">
那么，是否能够找到一个理论方法来真正的破解两军问题呢？答案是有的，量子通讯协议，笔者没有能力弄清这个颇为高深的问题。据我的理解，处于量子纠缠态的两个粒子，无论相隔多远都能够此同步，光是直观的来看，这个效应可以用来实现保密通讯。
但是由于测不准原理，一测量粒子状态就会改变其状态，所以通讯时还必须通过不可靠信道发送一条信息。尽管这个“另一条信息”是不可靠的，但是由于已经存在了一条绝对可靠的信道（量子纠），保证了另一条信道即使不可靠也能保证消息是可靠的，否则至少被窃取了一定能够被发现。
因此我们可以相信，至少理论上两军问题是可解的，即存在一种方法，即使利用了不可的信道，也能保证信息传递的可靠性。所以，在确保了信道可靠的基础上，我们可以回到拜庭将军问题上继续讨论。
 
<h2 id="toc_h2_3">2.问题实质及形式化</h2>
我们已经了解了拜占庭将军问题的场景，并且明确了这个问题的解决是建立在通信兵可以正确的达信息的基础上的，即信道绝对可信。同时，通过前文对于两军问题的探讨，我们明白了理论上可信信道也是可以实现的。接下来，我们将探讨拜占庭将军问题的实质。
<h3 id="toc_h3_4">2.1. 拜占庭将军问题实质</h3>
回顾问题，一群将军想要实现某一个目标（一致进攻或者一致撤退），但是单独行动行不通，必合作， 达成共识；由于叛徒的存在，将军们不知道应该如何达到一致。注意，这里“一致性”才是拜庭将军问题探讨的内容，如果本来叛徒数量就已经多到了问题不可解的地步，这个就是“反叛”的问了；同时，我们的目标是忠诚的将军能够达成一致，对于这些忠诚的将军来说，进攻或者撤退都是可的，只要他们能够达成一致就行。
但是，光靠“一致”就可以解决问题吗？考虑一下，如果万事俱备，客观上每个忠诚的将军只要攻了就一定能够胜利，但是却因为叛徒的存在他们都“一致的”没有进攻；反之，条件不利，将军们应该进攻，但是却因为叛徒的存在所有人都“一致的”进攻了。
可以发现，只有“一致性”是不足以解决拜占庭将军问题的，我们还需要提出一个“正确性”要。这个要求是值得斟酌的，因为如果客观来看或许会有“绝对正确的”判断，但是针对每一个将军，家的判断或许都不相同，我们如何定义“正确”呢？我们或许可以简单地说，正确就是每个忠诚的将都正确的表达了自己的意思，不会因为叛徒让别的将军认为忠诚的将军是叛徒而不采用他传达的消息
至此，我们将拜占庭将军问题简化成了，所有忠诚的将军都能够让别的将军接收到自己真实意图，并最终一致行动；而形式化的要求就是，“一致性”与“正确性”。
如果将问题推广开来，可以发现针对一致性和正确性的算法并不要求命令必须是“进攻/撤退”是“1/0”，而可以是“发送消息1/发送消息2/待机”或“x/y/z/w”，这意味着拜占庭将军问题算法以为多种分布式系统提供启发，比如电力系统或网络系统。
由此可见，这个问题说到底是一个关于一致性和正确性的算法问题，这个算法是针对的忠诚的将军，因为叛徒可以做出任何超出约定的判断。我们就是要在有叛徒的干扰下，找到一个抗干的算法。要解决这个算法问题，我们需要将形式化要求具体化。
<h3 id="toc_h3_5">2.2.形式化条件的推演</h3>
定义一个变量vi（为不失一般性，并不要求vi是布尔值），作为其他将军收到的第i个将军的命令；i将军会将把自己的判断作为vi。可以想象，由于叛徒的存在，各个将军收到的vi值不一定是相同的之后，定义一个函数来处理向量(v1,v2,...,vn)，代表了多数人的意见，各将军用这个函数的结果作为己最终采用的命令。至此，我们可以利用这些定义来形式化这个问题，用以匹配一致性和正确性。</>
1）一致性
条件1：每一个忠诚的将军必须得到相同的(v1,v2,...,vn)指令向量或者指令集合。
这意味着，忠诚的将军并不一定使用i将军送来的信息作为vi，i将军也可能是叛徒。但是仅靠这条件，忠诚的将军的信息送来的信息也可能被修改，这将影响到正确性。
2）正确性
条件2：若i将军是忠诚的，其他忠诚的将军必须以他送出的值作为vi。
如此，我们得到了一致性和正确性的形式化条件（条件1和条件2），这个条件是充分条件。考虑正确性条件是针对单个将军，而一致性条件是针对所有将军的，为方便我们重写一致性条件为

条件1’：无论i将军是忠诚或是叛徒，任何两个忠诚的将军都使用相同的vi。
条件1和条件1’是完全等价的。这是很巧妙的一步转换，如此一致性条件（条件1’）和正性条件（条件2）都只涉及一个将军i如何帮别的将军接受自己送出的值vi，所以可以将问题改为司令-官模式来简化问题，即一个司令把自己的命令传递给n-1个副官，使得：
IC1：所有忠诚的副官遵守一个命令，即一致性。
IC2：若司令是忠诚的，每一个忠诚的副官遵守他发出的命令，即正确性。
IC1和IC2分别由条件1’和条件2演化得来。司令-副官模式只要将司令遍历各个将军，就可以变成整问题，而他们采用的算法可以是完全一致的。IC1和IC2构成了解决拜占庭将军问题的充分条件，在种模式下，司令副官的形式下达成的一致意味着司令的命令得到了有效传达，若出现了异议，有异议将军会作为司令发起新的司令副官模式寻求自己的观点表达，以协商达成一致。
接下来，我们可以讨论拜占庭将军问题的算法了，这个算法只要能够满足IC1和IC2，就代表这个法可以切实有效的解决拜占庭将军问题。
在经典的情形下，我们可以找到两种办法，<code>口头协议</code>和<code>书面议</code>。笔者将会逐一探讨两种算法的推演和证明，其中证明部分并不会采用纯推理而以介绍证明思路为主。
事实上，若完整进行了算法推演，对算法已经能够有一个大致的了解。口头协议和书面协议会有多不同的启发，口头协议的实现起来简单，但是算法复杂，同时需要克服的困难更多；书面协议的算本身很简单，却能够克服很多问题，但是实现算法并不容易。这些不同让两者有了不同的使用场景和体实现。 
<h2 id="toc_h2_6">3.口头协议</h2>
首先，我们明确什么是口头协议。我们将满足以下三个条件的方式称为口头协议：
A1：每个被发送的消息都能够被正确的投递
A2：信息接收者知道是谁发送的消息
A3：能够知道缺少的消息
简而言之，信道绝对可信，且消息来源可知。但要注意的是，口头协议并不会告知消息的上一个源是谁。
先告知结论：采用口头协议，若叛徒数少于1/3，则拜占庭将军问题可解。也就是说，若叛徒数m，当将军总数n至少为3m+1时，问题可解（即满足了IC1和IC2）。
这个结论说明了，一个三模冗余的系统只能容故障冻结类型的错误，即一个元件故障以后就卡住动了（也即已知错误后会出现的结果），那么三模冗余是足够的。
但是对于拜占庭将军问题，由于叛徒可以做出各种各样的判断，就必须四模冗余系统才足够容错口头协议算法就是把自己的命令告诉其他人，并利用对其他人的命令取多数的方法来得到自己的结论但要注意的是，别的将军传来的命令是通过算法递归的方法来确定的。利用这个方法，可以保证在叛数量少于1/3的情况下，忠诚的将军可以实现一致性和正确性要求，即问题可解。
那么，口头协议算法又是怎么实现叛徒数少于1/3即可容错的呢？下面，笔者将介绍Lamport在论文中提出的口头协议OM(m)算法。笔者将会逐一介绍口头协议算法的详细内容、实例推演及证明这一部分将会需要您花一些时间来思考。
<h3 id="toc_h3_7">3.1.口头协议算法OM(m)</h3>
OM(0)算法
（1）司令将他的命令发送给每个副官。
（2）每个副官采用从司令发来的命令；如果没有收到命令，则默认为撤退命令。
OM(m)算法
（1）司令将他的命令发送给每个副官。
（2）对于每个i，vi是每个副官i从司令收到的命令，如果没有收到命令，则默认为撤退命令。副官在OM(m-1) 中作为发令者将之发送给另外n-2 个副官。
（3）对于每个i，和每个j ≠ i，vj 是副官i 从第2步中的副官j （使用OM(m-1)算法）发送过来的令，如果没有收到第2步中副官j 的命令，则默认为撤退命令。最后副官i 使用majority(v1,...,vn-1)得命令。
其中，majority(v1,...,vn-1)代表了大多数人的命令，若不存在则默认为撤退命令。
要一遍读懂这个算法并不容易，笔者也是花了不少时间研究这一小段文字才弄明白的。不过您不担心，笔者将会解释几个值得注意的点，并利用一个不难的实例来帮助您理解这个算法。
（1）算法始终保证了一个更加安全的默认值，这意味着若信息没有传到是可知的，并且传输时不在考虑范围内。
（2）这个算法是一个递归算法，在OM(m)中需要采用OM(m-1)得到相关结果。m代表的是叛

数量，从m到0，意味着对于每个将军，需要m+1轮的算法才能完成。
（3）该算法是关于m的，意味着使用该算法必须知道有多少个叛徒。或者说，利用该算法，可保证叛徒数量在某一个最大值（即总将军数量的1/3）之下时，拜占庭将军问题可解。
（4）对于任意k<m，在第m-k+1步中OM(k)的vi，都是利用OM(k-1)得来的，即每个将军将在OM(k-1)中询问其他人，i将军传给他们的是什么，而其他人传递回来的信息则是利用OM(k-2)得到
这个就是递归的意义，若您觉得笔者表达得不甚清楚，不用担心，您只用记住每一步都会牵涉到面很多步骤就可以了，之后将在实例推演中明白算法的核心思路。
 
<h3 id="toc_h3_8">3.2.口头协议实例推演</h3>
 首先，笔者将先举一个m=1，n=3的例子来说明三模冗余的问题所在，并介绍m=1，n=的时候系统是怎么容错的，这样您就可以明白算法是运行的。但由于m=1的时候并没有体现递归的过（因为m-1就变成了0），笔者将再举一个m=2，n=7的例子来说明算法递推的过程。 （1）m=1，n3的情形
n=3，意味着一个司令发送命令给两个副官，m=1意味着他们中有一个叛徒。 首先考虑司令忠而副官2是叛徒的情况。

司令把进攻命令传给了两个副官1和副官2，但是由于副官2为了不让他们达成一致，将司令的命改成了撤退。那对于副官1来说，他应该如何判断？他无法知道是司令是叛徒还是副官2是叛徒，从而法判断。

而如果司令是叛徒，两个副官忠诚，司令会发送两个不同的命令。当两个副官对照命令时，他们凌乱了，无法判断司令是叛徒或者对方是叛徒，从而又无法判断。这个情形非常简易的说明了三模冗是无法动态容错的。 （2）m=1，n=4的情形 n=4，意味着一个司令发送命令给三个副官，m=1意味他们中有一个叛徒。 首先考虑司令忠诚而副官3是叛徒的情况。

倘若司令在OM(1)中给各副官发送的消息都是进攻（A），之后OM(0)时，叛徒副官3给副官1和官2说他收到的消息是撤退（R）。那么对于副官1（或副官2）来说，他综合司令、副官3和副官2（副官1）后得到的消息向量都将会是(A,A,R)，利用majority函数之后，将会采用A，满足了IC1和IC2（span>回顾IC1：所有忠诚的副官遵守一个命令，IC2：若司令是忠诚的，每一个忠诚的副官遵守他发的命令）。

倘若司令是叛徒，那么我们已经不需要满足IC2。为方便，我们假设叛徒司令在OM(1)会给三个官发送的信息是(x,y,z)，其中x，y，z都可以是A或R的任意一种。之后，三位忠诚的副官将会按照OM()要求的那样，交换他们收到的信息。
对于副官1，他综合司令、副官2和副官3后得到的消息向量将会是(x,y,z)，可以发现对于其他两忠实的副官，他们得到的消息向量也将是(x,y,z)。不管x，y，z如何变化，<span>majority(x,y,z)对三人来说都是一样的，所以三个副官将会采用一致的行动</span>。
（3）m=2，n=7的情形
接下来，我们将讨论m=2，n=7的情形，虽然只是多了一个叛徒，但是这里会出现递归过程，所会复杂很多。
首先，我们先讨论司令忠诚的情形，假设叛徒为副官5和副官6。

.com/images/img-loading.svg" alt="m=2，n=7中司令忠诚而副官5和副官6是叛徒的情形" width="548" height="153" data-src="http://7fvhfe.com1.z0.glb.clouddn.com/wp-content/uploads/215/10/%E5%9B%BE8.png"></a>
在OM(2)中，司令将攻击命令（A）传给各个副官。在OM(1)中，忠诚的副官们将会发送他们收的消息（A），但由于副官5和副官6是叛徒，他们将会发送别的信息（比如R）。这时，忠诚的副官将会采用使用OM(1)中的方法来确定各个v1~v6。例如，对于副官1，他收到了司令传来的命令，他会直接采用majority函数综合司令和其他将军传来的信息吗？他不会，因为这还在OM(1)中，他不知道司令是不是叛徒，他会利用询问别人的方式来确认将军的命令，但是按照算法他会把令的命令作为v1（即v1=A）并传给其他人。
接下来他会努力取得其他的v2~v6的值，这时已经在OM(1)中了，副官1绝不会轻易相信别人传的消息，比如副官2给他传来了命令A，但是他会怀疑副官2传来的消息，所以他用OM(1)大法，问其人副官2传给了他们什么，副官3和副官4诚实的告诉副官1，副官2给他们传的是A，而这时副官5和副6又要撒谎了，他们又乱说，我们姑且假定他们传来的是x’和y’吧。这样，终于进入到了OM(0)，时副官1将会综合其他副官对于v2的反馈，得到向量(A,A,A,x’,y’)，再利用majority函数，得到了v2A。如下图，这是副官1在OM(1)中得到的信息（x，y等表示了不确定的命令）。
<img title="司令忠诚时副官1在OM(1)中得到的信息" src="https://static.hacpai.com/images/mg-loading.svg" alt="司令忠诚时副官1在OM(1)中得到的信息" data-src="http://www.btckan.co/public/resources/pic/news/2015/10/16/M3hn4R2PHu.png">我们就可以得到副官1的v1~v6向为(A,A,A,A,x,y)，利用majority函数，副官1最终采用的行动会是A。 类似的，我们可以发现其他几个忠诚的副官得到的命令向量都会是(A,A,A,A,x,y)，利用majority函数后采的行动都会是A。所以，我们可以发现忠诚的副官采用的命令都是A（满足IC1），并且和忠的将军的命令一致（满足IC2）。至此，您应该已经明白了这个算法是怎么递归的，不管m等于多少都只是一个算法步骤多寡的问题。 至于司令是叛徒的情形，其实是相似的，这里简单的再提一下便于解。若您已经明白了算法过程，完全可以跳过。
<img title="m=2，n=7中司令和副官6是叛徒的情形" src="https://static.hacpai.com/images/mg-loading.svg" alt="m=2，n=7中司令和副官6是叛徒的情形" data-src="http://www.btckan.com/public/resources/pic/news/2015/10/16/Ry0Y4G8qsU.png">
为方便，我们假定了副官6是叛徒。司令在OM(2)中就很鸡贼的给副官1~副官6发送了不同的命令A,R,A,R,A,x)。在OM(1)中，各副官把自己收到的命令传出去，而（为方便，假定）副官6则给其他副分别发送了(A,R,A,R,A)。类似于前文推演的那样，考虑副官1，他将司令传来的命令A作为v1后，还询问其他人传来的命令，由此去确认v2~v6，类似的，我们将之表达为下图：
<img title="司令反叛时副官1在OM(1)中得到的信息" src="https://static.hacpai.com/images/mg-loading.svg" alt="司令反叛时副官1在OM(1)中得到的信息" data-src="http://www.btckan.co/public/resources/pic/news/2015/10/16/H7UZ5tmjVk.png">
如图，我们就可以得到副官1的v1~v6向量为(A,R,A,R,A,A)，利用majority函数，副官1最终采用行动会是A。类似的，我们可以发现忠诚的副官1~副官5得到的消息向量都是(A,R,A,R,A,A)，最终他采用的行动都会是A（满足了IC1），而司令是叛徒不需要满足IC2。值得提醒的是，若副官6传递的是R,A,R,A,R)，那么他们所有人得到的消息向量都是(A,R,A,R,A,R)，此时A和R数量一样多，这并不代表ajority不起作用了，他将采用默认值R（回顾前文：majority(v1,...,vn-1)代表了大多数人的令，若不存在则默认为撤退命令），所有人的行动都会采用R，这同样是满足的。
到此为止，我们已经将口头算法的实例推演进行的很彻底了，若您还有兴趣，可以试一试当n=7m=3的时候为什么就不能达成一致了，操作是类似的。
<h2 id="toc_h2_9">3.3.口头协议算法证明</h2>
算法的证明思路其实并不复杂，简单的来说，对于一个递归算法，我们使用数学归纳法来证明是直观而又有效的方法了。我们回顾一下命题：将军总数为n，叛徒数量为m，OM(m)可以实，在n>3m的情况下，使得：
IC1：所有忠诚的副官遵守一个命令。
IC2：若司令是忠诚的，每一个忠诚的副官遵守他发出的命令。
为了证明整个命题，我们先引入一个针对IC2的引理：
引理：对于任意m和k，如果有超过2k+m 个将军和最多k 个背叛者，那么算OM(m)满足IC2。
证明：
（1）m=0，而将军是忠诚的，直接满足IC2；
（2）m>0，此时假定OM(m-1)是有效的，那么只需要考虑OM(m)这一轮即可。</s

an>
n>2k+m，意味着n-1>2k，n-1是除了司令以外的所有副官，而所有副官的数量比叛徒的倍还多，那他们直接利用majority函数的时候，就可以直接正确得到司令的命令。
可以发现，这个引理说明了如果只需要考虑IC2，将军总数是不需要3倍背叛者之多的，接下来我回归命题。
证明：
首先考虑司令是忠诚的，令引理中的k=m，直接得到OM(m)可以满足IC2。
这时，我们只用考虑司令是叛徒的状况。同样利用数学归纳法。
（1）m=1，之前我们已经推演过OM(1)可以满足1个叛徒司令，3个忠诚副官的情况；
（2）m>1，那么假设n’ >3m’ 的情况下，OM(m-1)能够满足IC1和IC2。
由于司令是叛徒，在OM(m)中司令会把命令发给各个副官，而这些副官中会有m-1个叛徒。在一轮中，副官的数量至少有3m个，叛徒数为m-1，很显然3m>3(m-1)，也就是说n’ >3m’ 根据假设，OM(m-1)可以满足IC1和IC2，尽管司令是叛徒，由于OM(m-1)是有效的，OM(m)这一轮忠诚的副官可以得到相同的(v1,…,vn-1)向量，所以忠诚的副官将会利用majority函数采用相同的命令得证。
总结一下，口头协议中，我们始终要求的是相同的(v1,…,vn-1)向量，只要这个向量是相同的我们么处理都可以。又由于算法是递归的，所以我们一定需要把这个处理方法变得通用而逻辑有效才行，以我们才选用了majority函数这个算法。这一点至关重要却又没有这么直观，因为我们的第一反应是得到相同的majority函数值。但是反过来一想，既然算法是递归的，majority函数值相同不就意味着(v,…,vn-1)向量相同吗？正确理解递归的思想是使用该算法和利用数学归纳法证明该算法的关点。
至此，我们已经大致明确了口头协议是怎么回事，可以做到什么不能做到什么，以及这个算法的演和证明。很多系统都会出现口头协议的情况，即各个系统节点可以把自己的消息准确的发送出去，时可以知道消息的来源于何处。但是，这个方法的消息并不能追本溯源，这使得在口头协议中至少得模冗余，我们可以试图找到一个方法，让消息能够追本溯源，可以想象这能够拓宽使用条件，这个方可以是书面协议。
 
<h2 id="toc_h2_10">4.书面协议</h2>
口头协议中我们讨论了很多，揭示了口头协议的缺点是消息不能追本溯源，这使得口头协议必须四模冗余的情况下才能保证正确。但是，若能引入一种方法让消息能够追本溯源，情况会不会有所改呢？这就是书面协议引入的灵感。
除了A1，A2和A3以外，我们在口头协议之上添加一个条件A4，使之成为书面协议（A1：每个发送的消息都能够被正确的投递；A2：信息接收者知道是谁发送的消息；A3：能够知道缺少的消息）/p>
A4：（a）签名不可伪造，一旦被篡改即可发现，而叛徒的签名可被其他叛徒伪造；（b）任何都可以验证签名的可靠性。
那么，我们先说结论：对于任意m，最多只有m个背叛者情况下，算法SM(m)能解决拜占庭将军题。也就是说，在使用签名的情况下，书面协议可以打破三模冗余的僵局，使用了签名的情况下，只知道了叛徒数量，我们就可以利用SM(m)算法解决拜占庭将军问题。 
<h3 id="toc_h3_11">4.1.书面协议算法SM(m)</h3>
口头协议算法我们已经讨论过很多了，所以笔者对书面协议尽量简短的介绍。回顾
IC1：所有忠诚的副官遵守一个命令，即一致性。
IC2：若司令是忠诚的，每一个忠诚的副官遵守他发出的命令，即正确性。
我们要找到一个算法SM(m)，不管将军总数n和叛徒数量m，只要采用该算法，忠诚的将军总能到一致（满足IC1和IC2）。我们用集合Vi来表示i副官收到的命令集，这是一个集合，也就是满足互异（没有重复的元素）等集合的条件。类似的，我们定义choice(V)函数来决定各个副官的选择，这个数可以有非常多种形式，他只要满足了以下两个条件：
（1）如果集合V只包含了一个元素v，那么choice(V)=v
（2）choice(o)=RETREAT，其中o是空集
任何满足了这两个条件的函数都可以作为choice()，例如取平均值就可以。我们只需要根据具体形定义choice()即可，这个非重点，笔者并不加以讨论，您可以自行思考。之后我们会发现SM(m)算并不是一个递归算法，我们只要让各个副官收到的V集相同，choice(V)也一定能够得到相同的值。</>
简单解释该算法如下：

初始化Vi=空集合。
（1）将军签署命令并发给每个副官；
（2）对于每个副官i：
（A）如果副官i从发令者收到v:0的消息，且还没有收到其他命令序列，那么他
（i）使Vi为{v}；
（ii）发送v:0:i给其他所有副官。
（B）如果副官i收到了形如v:0:j1:...:jk的消息且v不在集合Vi中，那么他
（i）添加v到Vi；
（ii）如果k<m，那么发送v:0:j1:...:jk:i 给每个不在j1,..,jk 中的副官。
（3）对于每个副官i，当他不再收到任何消息，则遵守命令choive(Vi)。
值得注意的是，如果司令忠诚，由于其签名不可伪造，所有忠诚的副官都将得到一个单点集{v}，们采用的命令集Vi相同，得到的choive(Vi)也为v，满足了IC1和IC2。
如果司令并非忠诚，只需要满足IC1，但是算法SM(m)使得所有忠诚的副官得到相同的Vi，使用coice()函数后采用的命令也就一定相同。
<h3 id="toc_h3_12">4.2.书面协议实例推演</h3>
司令是叛徒的状况稍难想象，举个例子，n=3，m=1，其中司令是叛徒，这是口头协议不能解决状况。

很显然，副官1得到的V1={A,R}，副官2得到相同的V2={A,R}。他们采用choice函数后得到的命一定相同。
相似的，n=4，m=2，其中司令是叛徒，这同样是口头协议不能解决的状况。

 
副官1和副官2得到的V1=V2={A,R}，他们采用choice函数后得到的命令也相同。即使命令不是尔值，经过上面的分析框架，也可以得到相似的结论。至于这个算法的证明，有兴趣的读者可以参考Lmport的原文。
(https://www.microsoft.com/en-us/research/wp-content/uploads/2016/12/The-Bzantine-Generals-Problem.pdf）
书面协议的本质就是引入了签名系统，这使得所有消息都可追本溯源。这一优势，大大节省了成，他化解了口头协议中1/3要求，只要采用了书面协议，忠诚的将军就可以达到一致（实现IC1和IC2。这个效果是惊人的，相较之下口头协议则明显有一些缺陷。
书面协议的结论非常令人兴奋，这不是解决了拜占庭将军问题了吗？但请注意我们在A1~A4中际上是添加了一些条件的，这使得拜占庭将军问题在这些假设下能够解决，但是在实际状况中却会有些问题。观察A1~A4，我们做了一些在现实中比较难以完成的假设，比如没考虑传输信息的延迟时间书面协议的签名体系难以实现，而且签名消息记录的保存难以摆脱一个中心化机构而独立存在。事实，存在能够完美解决书面协议实际局限的方法，这个方法就是区块链。
 
 