# 问题描述

拜占庭将军问题是一个共识问题，由Lamport与另外两个人在1982年提出。问题的核心是军中可能有叛徒，却要保证进攻一致，由此引申到计算机邻域，发展成了一种容错理论。

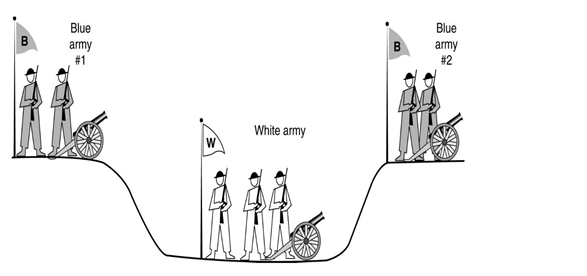
拜占庭帝国想要进攻一个强大的敌人，为此派出了10支军队去包围这个敌人。这个敌人虽不比拜占庭帝国，但也足以抵御5支常规拜占庭军队的同时袭击。基于一些原因，这10支军队不能集合在一起单点突破，必须在分开的包围状态下同时攻击。他们任一支军队单独进攻都毫无胜算，除非有至少6支军队同时袭击才能攻下敌国。他们分散在敌国的四周，依靠通信兵相互通信来协商进攻意向及进攻时间。困扰这些将军的问题是，他们不确定他们中是否有叛徒，叛徒可能擅自变更进攻意向或者进攻时间。在这种状态下，拜占庭将军们能否找到一种分布式的协议来让他们能够远程协商，从而赢取战斗？这就是著名的拜占庭将军问题。

需要明确的是，在拜占庭将军问题中并不考虑通信兵是否会被截获或无法传达信息等问题，即消息传递的信道是没有问题的。**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到一致性是不可能的。**所以在研究这个问题的时候，我们已经假定信道是没有问题的了，并在这个前提下，做**一致性和容错性**的相关研究。

对于信道有问题的情况，这涉及到另一个相关问题：两军问题。

## 两军问题

两军问题和拜占庭问题实质是不一样的，并且问题的前提和研究方向都截然不同。



**两军问题**

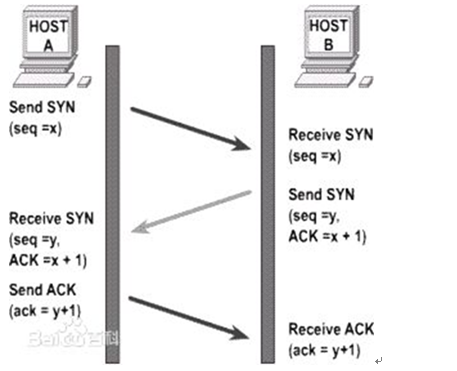
如上图所示，白军驻扎在沟渠里，蓝军则分散在沟渠两边。白军比任何一支蓝军都更为强大，但是蓝军若能同时合力进攻则能够打败白军。他们不能够远程的沟通，只能派遣通信兵穿过沟渠去通知对方蓝军协商进攻时间。是否存在一个能使蓝军必胜的通信协议，这就是两军问题。

在两军问题中，需要注意的是通信兵经过敌人的过程中可能会被捕，也就是说在两军问题中**信道是不可靠的，但是并没有叛徒之说。**这就是两军问题和拜占庭将军问题的根本性不同。

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，两军问题就可以解决了。但是并不存在这样一种信道，所以两军问题没法解决，比如1号给2号派出了通信兵，如果1要确认2是否收到自己的信息，1只能要求2在收到之后给自己一个回执。2在发出回执的时候也没法确定1是否收到自己的回执，所以要求1在收到之后给2发一个回执，同样1发出后也必须收到2的回执…

在这样一个系统中，永远需要存在一个回执，这里还没有考虑通信兵修改通信内容的问题，所以不存在一个能使蓝军一定胜利的通信协议。

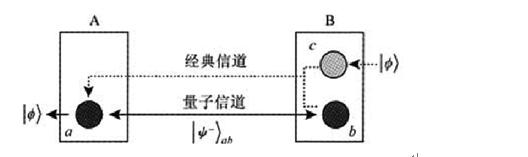
但是两军问题在现代通信系统中必须解决，如果不能解决，那么意味着一方发出的信息可能会出现丢失、篡改等情况。是否可以通过一种相对可靠的方式来解决大部分情况呢？



在TCP协议中，A向B发出一个随机数x，B收到x之后，发送另一个随机数y以及x+1作为答复，这样A就确定B收到了，因为要破解随机数x的可能性不大；然后A再发送y+1给B作为答复，这样B就知道A已经收到了。这样A和B之间就建立了一个可靠的连接，彼此相信对方已经收到并确认了信息。

事实上A并不知道B是否收到了y+1，并且由于信道的不可靠，x或者y都是可能被截获的，这些问题说明即使是三次握手，也并不能彻底解决两军问题，只是在现实成本可控的条件下，我们把TCP协议当作两军问题的现实可解方法。

理论上可以采用量子通讯协议解决两军问题，因为处于量子纠缠态的两个粒子，无论相隔多远都能够彼此同步，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另一条信息。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（量子纠缠），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。



# 拜占庭将军问题

我们已经明确这个问题是建立在通信兵可以正确传达信息的基础上，即信道绝对可信。

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，必须合作达成共识。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”才是拜占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”的问题了；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以的，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进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，将军们不应该进攻，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”要求。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，大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军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的真实意图，并最终一致行动**，而形式化的要求就是，“一致性”与“正确性”。

如果把问题推广开，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”或者是“1/0”，可以是“消息1/消息2/消息3”，这意味着拜占庭将军问题算法可以为多种分布式系统提供解决办法。

**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，这个算法是针对的是忠诚的将军，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，找到一个抗干扰的算法。要解决这个算法问题，我们需要将形式化要求具体化。**

## 问题的推演

变量vi作为其他将军收到的第i个将军的命令值，由于叛徒的存在，各个将军收到的收到的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)，选择代表多数人的意见，各将军把函数的结果作为自己最终采用的命令。

1. 一致性

每个忠诚的将军必须得到相同的(v1,v2,…,vn)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。

但是仅靠这个条件，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。

1. 正确性

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。考虑到**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′）和正确性条件（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-副官模式来简化问题，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**的副官都遵守一个命令，即一致性；

IC2：**若司令是忠诚的**，每个忠诚的副官必须遵守他发出的命令，即正确性。

司令-副官模式只要将司令遍历各个副官，就可以变成完整问题，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，在这种模式下，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，有异议的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。

基于此，我们考虑拜占庭将军问题的算法，这个算法只要满足IC1和IC2，就代表这个算法可以切实有效的解决拜占庭将军问题。有两种方法，**口头协议**和**书面协议。**

口头协议和书面协议会有很多不同的启发，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；书面协议的算法本身很简单，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具体实现。

## 口头协议

我们将满足以下三个条件的方式称为口头协议：

1. 每个被发送的消息都能够被正确的接收；
2. 信息接受者知道谁发送的消息，但不知道上一个来源；
3. 能够知道缺少的消息。

采用口头协议，如果叛徒数小于1/3，则拜占庭问题可解。也就是说，如果叛徒数为m，将军总是大于等于3m+1时，问题可解。

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，可以保证在叛徒数量少于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。

### 口头协议算法OM(m)

**OM(0)**

1. 司令把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，默认采用撤退命令。

**OM(m)**

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i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）发送过来的命令，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)得到命令。

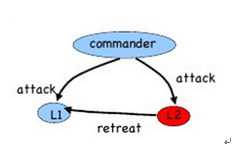
majority(v1,…,vn-1)代表了大多数人的命令，如果不存在采用默认撤退命令。

这个算法中需要注意的一些点：

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，并且传输时间不在考虑范围内。
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代表的是叛徒数量**，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，可以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，即每个将军将会在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。

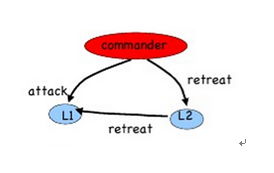
### 口头协议一个实例

首先看一个m=1，n=3的例子，意味着一个司令发送命令给两个副官，m=1说明其中有一个时判读。假设副官2是叛徒。

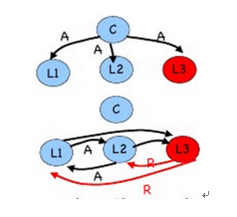


司令把进攻命令传给了两个副官1和副官2，但是由于副官2为了不让他们达成一致，将司令的命令改成了撤退，在OM(0)阶段，副官2发送了撤退给副官1，那对于副官1来说，他应该如何判断？他无法知道是司令是叛徒还是副官2是叛徒，从而无法判断。

同样如果司令是叛徒，两个副官是忠诚的，司令会发送两个不同的命令给两个副官，两个副官任然无法判断司令是叛徒还是另一个副官是叛徒。



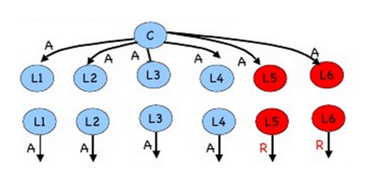
m=1，n=4的情形，先假设司令忠诚，副官3是叛徒。



假设OM(1)司令给个副官发送的消息都是进攻，到OM(0)时，副官3会告诉副官1和副官2说他收到的消息是撤退。副官1在综合他收到的司令、副官2和副官3的消息将会得到(A，A，R)，majority后会采用A，这样就满足了LC1（所有忠诚的副官遵守一个命令，因为每个副官最终都会采用A）和LC2（若司令是忠诚的，每一个忠诚的副官遵守他发出的命令，这里的司令是忠诚的，而且也采用了他发出的命令）。

如果司令是叛徒，这是不再需要考虑LC2了，假设叛徒司令在OM(1)阶段发送个三个副官的命令是(x,y,z)，三个忠诚的副官在OM(0)阶段会交换他们收到的信息，这些副官在经过OM(0)之后，收到的都是(x,y,z)，执行了majority之后三个人都是一样的，所以符合LC1.

对于m=2，n=7时，先假设司令是忠诚的，副官5和副官6是叛徒。



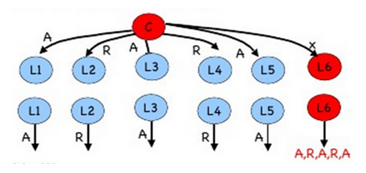
OM(2)中，司令会发送A给各个副官，OM(1)，副官5和副官6会告诉其他副官他们收到的是R，其他忠诚的副官发送的是A，这时，忠诚的副官们将会采用使用OM(1)中的方法来确定各个v1~v6。例如，对于副官1，他收到了司令传来的命令，他会直接采用majority函数综合司令和其他将军传来的信息吗？他不会，因为这还在OM(1)中，他并不知道司令是不是叛徒，他会利用询问别人的方式来确认将军的命令，但是按照算法他会把司令的命令作为v1（即v1=A）并传给其他人。

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来的消息，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，问其他人副官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和副官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)，这时副官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v2=A。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）。



我们就可以得到副官1的v1~v6向量为(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函数后采用的行动都会是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），并且和忠诚的将军的命令一致（满足IC2）。

对于司令是叛徒的情形，假设司令和副官6是叛徒，司令在OM(2)给副官发送命令(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则给其他副官分别发送了(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，还会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：



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最终采用的行动会是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)，最终他们采用的行动都会是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majority不起作用了，他将采用默认值R（回顾前文：majority(v1,…,vn-1)代表了大多数人的命令，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。

### 口头协议算法证明

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最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)可以实现，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，那么算法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。

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两倍还多，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，接下来我们回归命题。

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。在下一轮中，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)这一轮中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怎么处理都可以。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，所以我们才选用了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要得到相同的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(v1,…,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键点。

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，以及这个算法的推演和证明。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，同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四模冗余，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，这个方法可以是书面协议。

## 书面协议

口头协议的缺点就是消息不能追本溯源，使得口头协议必须在四模冗余的情况下才能保证正确，引入书面协议就是为了让消息能够追本溯源。

书面协议在口头协议基础上增加了一个条件

1. 每个被发送的消息都能够被正确的接收；
2. 信息接受者知道谁发送的消息，但不知道上一个来源；
3. 能够知道缺少的消息。
4. 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；任何人都可以验证签名的可靠性。

**对于任意m，最多只有m个背叛者情况下，算法SM(m)能解决拜占庭将军问题。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，只要知道了叛徒数量，我们就可以利用SM(m)算法解决拜占庭将军问题。**

### 书面协议算法OM(m)

在口头协议中，

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，忠诚的将军总能达到一致（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，也就是满足互异性（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，这个函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情形定义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)算法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。

解释该算法：

初始化Vi=空集合

1. 将军签署命令并发给每个副官；
2. 对每个副官i:

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：

使Vi为{v}；

发送v:0:i给其他所有副官。

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

添加v到Vi；

如果k<m，那么发送v:0:j1:…:jk:i给每个不在j1,…,jk中的副官。

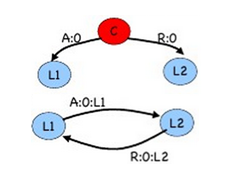
1. 对于每个副官i，当他不在收到任何消息，这遵守命令choive(Vi)

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}，他们采用的命令集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choice()函数后采用的命令也就一定相同。

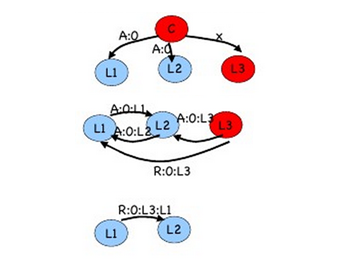
### 书面协议一个实例

当n=3，m=1是，口头协议是没法解决的，在书面协议中，



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函数后得到的命令一定相同。

相似的，n=4，m=2，其中司令是叛徒时，



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。即使命令不是布尔值，经过上面的分析框架，也可以得到相似的结论。

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，大大节省了成本，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中实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一些问题。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。事实上，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。

# 拜占庭将军论文

将军之间只能通过信使传递信息，他们必须在相同的作战方案上达成一致。但是，他们之中可能存在叛徒，这些叛变的将军会竭力的扰乱其它人。拜占庭将军问题，就是指找到一种算法可以保证那些忠诚的将军可以达成一致。结果表明，如果使用口头信息，当且仅当超过2/3的将军是忠诚的时候该问题才可解，也就是说1个将军可以扰乱2个将军；如果使用不可伪造的书面信息，对于任何数目的将军和叛徒，该问题都是可解的。此外，本文还讨论了如何将该问题的解应用于可靠的计算机系统实现中。

一个可靠的计算机系统必须能够处理一个或多个的组件的失败。**一个失败的组件可能会表现出一种经常被忽略的行为：向系统的其他部分发送相矛盾的信息。**可以将处理这种失败的情况的问题抽象出来，就是这里的拜占庭将军问题。

有些将军可能是叛徒，他们会尽力阻止那些忠诚的将军达成一致，将军们必须有一个算法来保证如下条件：

A：所有忠诚的将军必须达成相同的行动计划；

忠诚的将军将会做该算法要求他们做的事情，但是叛变的将军可以做任何他们想做的事情。无论叛变的将军会做什么，算法必须要保证条件A。

诚实的将军不能仅仅达成一致，他们还应该达成一个合理的行动计划。也就是说，实际上我们想保证：

B：当只有少数人是叛徒时，他们不能导致那些忠诚的将军采纳一个糟糕的计划。

B很难去形式化，因为它需要精确的定义何为糟糕的计划，当然我们也并不尝试去给出这样的一个定义。我们来考虑将军们如何做出决定，每个将军都会观察敌情，并将他的观察结果告诉其他将军。假设**v(i)代表第i个将军发送的信息。**每个将军使用某种方法来根据这些信息v(1),v(2)……v(n)来拟定作战计划(n代表将军的总数)。通过让所有的将军使用同一种方法就可以满足条件A，通过使用一种健壮的方法条件B也可以满足。比如，现在需要决定是进攻还是撤退，v(i)代表第i个将军关于进攻还是撤退的意见，最终的决定可以通过在他们之间进行一个多数决的投票来决定。在这种情况下，只有当持两种意见的忠诚将军数目几乎相同时，少数的叛变将军才能影响最终的结果。但是这种情况下，无论是进攻还是撤退都算不上是糟糕的方案（这就说明满足条件B）。

虽然这种策略可能不是满足条件A和B的唯一一种方式，但是目前我们仅想到这一个。该方法假设存在一种方法，将军们可以相互传递各自的v(i)值。很明显的一种方法是，将军i，让他的信使将v(i)送给所有的将军。但是，这样行不通，因为**如果要满足条件A，需要每个忠诚的将军收到相同的v(1),v(2)……v(n)，但是一个叛变的将军可能会给不同的将军发送不同的值。**对于条件A来说，如果要满足下面的条件必须成立：

1. 每个忠诚的将军必须收到相同的v(1),v(2)……v(n)

条件1暗示**一个将军并没有必要使用一个直接从第i个将军那收到的v(i)值，**因为一个叛变的第i将军可能给不同的将军发送不同的v(i)值。但是这样意味着，满足条件1的同时，很可能一不小心就使用了一个与第i将军发送的v(i)不同的值，即使第i个将军是诚实的（因为我们可能采用了从其他将军处得来的关于v(i)的值，但是这个将军可能是叛变者，它可能自己已经改变了v(i)的值，这样即使第i将军是诚实的，但是经过叛变者之后它的值也已不再受控了）。但是为了满足条件B，我们绝不允许这种情况发生。比如我们，我们不能允许少数的叛变者就使得忠诚的将军们在一个个“retreat”, “retreat”…… “retreat”中做决定，而每个忠诚的将军发送的明明是“attack”。因此，我们还要为每个i增加如下的需求：

1. 如果第i个将军是忠诚的，那么其他的忠诚的将军必须使用他发送的值作为v(i)的值。

我们可以改写条件1，使得它是针对任意i的条件(无论第i个将军是否是忠诚的)：

1'. 任意两个忠诚的将军使用相同的v(i)值.

条件1'和2现在都是针对第i个将军的值v(i)的了。因此，我们可以只考虑一个将军如何发送他的值给其他人。现在我们用一个发令将军向它的下属发送命令的形式来重新描述这个问题，就得到如下问题：

拜占庭将军问题。一个发令将军向他的n-1个下属将军发送命令，使得：

IC1. 所有忠诚的下属都遵守相同的命令。

IC2. 如果发令将军是忠诚的，那么每个忠诚的下属必须遵守他发出的命令。

条件IC1和IC2被称为交互一致性(interactive consistency)条件。可以看到，当发令将军是忠诚的时候，IC1可以由IC2导出。然而，发令将军不一定是忠诚的。

为了解决最初的问题，第i个将军只需要利用拜占庭将军问题的解法发送命令”使用v(i)作为我的值”，就可以将他的值v(i)发送给其他的将军。此时其他的将军就扮演拜占庭将军问题中的下属的角色。

拜占庭将军问题看似很简单。它的难解之处是通过一个令人惊讶的事实而体现出来的：如果将军们只能发送口头消息，除非有超过三分之二的将军是忠诚的，否则该问题无解。尤其是，如果只有三个将军，其中一个是叛变者，那么此时无解。口头消息是指信息的内容完全在发送者控制之下，这样一个叛变了的发送者可能会传送任何可能的消息。正常情况下计算机之间传输的消息就属于这种类型。

# 参考

拜占庭将军问题深入探讨

<http://www.8btc.com/baizhantingjiangjun>

The Byzantine General Problem

<http://duanple.blog.163.com/blog/static/7097176720112643946178/>

论文

<https://dl.acm.org/citation.cfm?id=357176>

<http://delivery.acm.org/10.1145/360000/357176/p382-lamport.pdf?ip=219.228.135.198&id=357176&acc=PUBLIC&key=BF85BBA5741FDC6E%2E035EACC12F524219%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1516800573_809a72efd4c0591d65981dd3a7cb2cb6>