# Two Generals' Paradox

Craig West & Grégory Essertel

May 6, 2016

**Abstract**

The two general problem is often stated as follow: Two armies, each led by a general, are preparing to attack an enemy. The two armies are on a different flank of the enemy's army, and on there own, one army can not win against the enemy army, they must attack at the same time. In order to come to an agreement they must only send messengers, and have to cross enemies line. The paradox comes from the fact that the two generals can not be certain that they both agree to attack at the same time.

We are going to present a probabilistic solution to this issue and analyze its security properties.

## 1 Introduction

At first sight this problem doesn't seem to be difficult to solve. One general sends a message: "Attack at dawn", but the messenger may get captured during the expedition, and the second general needs to acknowledge the message: "OK attack at dawn". Unfortunately the last general to send the message can not be **certain** that the other general received the message, and sending an more confirmation messages leads to the same issue; this is why it is called a paradox. This problem has been the first computer communication problem to be proved to be unsolvable [1].

## 2 Fault tolerant systems

A fault tolerant system is a system designed for high availability. The principle idea is to handle situations where there are issues in communication. This kind of situation can happen when the communication medium between two hosts is not reliable, and a message gets through with a probability $p$. In this situation, the first general must send enough messengers to be *almost* certain that the other general got the message. The first general would like to have a probability to fail lower then $\epsilon > 0$, how many messages $N$ should he send?

$$\begin{aligned}(1-p)^N &< \epsilon \\ N\log(1-p) &> \log\epsilon \\ N &> \frac{\log\epsilon}{\log(1-p)}\end{aligned}$$

With $\epsilon = 10^{-5}$ and a probability of failure $p = 0.5$, the general needs to send $N = \frac{5}{\log 2} \approx 17$ messages.

This solution works well, but unfortunately in the presence of an attacker with this model doesn't hold. The attacker can choose to let the packet go or not. In the rest of the paper, we are modeling the problem as a Bank and an ATM, they must agree to actually perform a transaction. A customer would not like it if the Bank takes money from the account but the ATM doesn't dispense money. On the other hand the Bank doesn't want the ATM to dispense money without actually taking it from an account. The enemy army is a man in the middle that has full control of the traffic.

## 3 Our solution

As we stated before, this problem has been proven unsolvable. We want to find a probabilistic solution, and be able to evaluate its security. The previous solution didn't work in the case of a smart attacker. Indeed the attacker could choose to stop all messages, and then fool the first general. This means that the messages need to be related instead of independent. Our solution is the following: on top of an encrypted and authenticated channel (also all messages have a sequence number)

1. The Bank and ATM exchanged messages. It can be any number of messages.

2. The Bank or the ATM generates a random number $N$ based on a probability distribution defined before hand, and share it with the other side.

3. Both parties start the **commit protocol**:

- 1 - They both send a single message to the other party, and wait for the other party's message, or fail on a timeout.
- ...
- N - They both send a single message to the other party, and wait for the other party's message, or fail on a timeout. If it receives the $N^{th}$ message this party concludes an agreement.

## 4 Analysis

### 4.1 Proof of correctness

With this protocol, if the attacker modifies or drops one of the commit messages one party will detect the tampering and stop the protocol immediately. Therefore the other party will timeout and abort as well. The only packet that can lead the attacker to succeed is one of the $N^{th}$ messages, indeed like the protocol is over at the $N^{th}$ message even by detecting the attack there is nothing else it can do, it has to stop because it was the end of the protocol and stop because there was an attack. Therefore the other side will never beware of the attack.

### 4.2 Security

The attacker doesn't know the value of $N$, therefore, we assume the best strategy is to pick a random number $M$ and drop the $M^{th}$ commit message hopping that $N = M$.

In order to evaluate the security, we are going to use two different probability distributions: uniform and geometric. The parameters are going to be chosen in order to ensure that the numbers are picked mainly in the range $0, MAX\_COMMIT$. Also in our analysis we assume the attacker coerced the Bank and ATM to obtain the parameters used during communication.

**Uniform** distribution with a parameter $p = \frac{1}{MAX\_COMMIT}$, the chance that both the bank and the attacker choose the number $n$ is $p^2$. Therefore the chance for the attacker to succeed is $\sum_{n=1}^{MAX\_COMMITS} p^2 = p$. Which mean the attacker should succeed with a probability $p$.

**Geometric** distribution with a parameter $p = 1 - 10^{\frac{-3}{MAX\_COMMIT}}$. The parameters is chosen so that 99.9% of the values will be in the range $[0, MAX\_COMMIT]$. For this distribution, that chance that both choose the number $k$ is $p^2(1-p)^{2k}$, therefore the chance that the attacker guess correctly is $\sum_{k=0} p^2(1-p)^{2k} = p^2\frac{1}{1-(1-p)^2} = \frac{p}{2-p}$.

The two probabilities are not easily comparable as stated , we need to compare them with the common parameter $MAX\_COMMIT$. It is possible to show that $\forall x \geqslant 1, f(x) = \frac{1}{x} - \frac{1-10^{-3/x}}{1+10^{-3/x}} \leqslant 0$. Therefore the uniform distribution seems to be the best choice based on security. However the expected values are much smaller for the geometric distribution, which means that performance wise this is a better choice.

Table 1: Distribution comparison

| $MAX\_COMMIT$ | 10 | | $10^2$ | |
|---|---|---|---|---|
| Distribution | Geometric | Uniform | Geometric | Uniform |
| Attacker success | 33% | 10% | 3.4% | 1% |
| Expected value | 2 | 5 | 15 | 50 |
| $MAX\_COMMIT$ | $10^3$ | | $10^4$ | |
| Distribution | Geometric | Uniform | Geometric | Uniform |
| Attacker success | 0.34% | 0.1% | 0.035% | 0.01% |
| Expected value | 145 | 500 | 1448 | 5000 |

### 4.3 Evaluation

During our analysis we ran tests on different combinations of geometric and uniform distributions. We wanted to see how the attackers odds would change based upon which distribution the Bank and ATM were using compared to the distribution selected by the attacker.

Due to the amount of time we had and the length of time the experiments took to run, we did not collect all the information we would like. We would like to collect more results will larger commit numbers, however, they require a significant amount of time. The amount of time was primarily due to the added overhead of the protocol because we send a significant number of extra messages.

Table 2: Commit 10 - Successful attacks out of 1000 rounds

| Distribution | Geometric | Uniform |
|---|---|---|
| Geometric | 30.4% | 7.7% |
| Uniform | 7.5% | 8.7% |

Table 3: Commit 10 - Successful attacks out of 10000 rounds

| Distribution | Geometric | Uniform |
|---|---|---|
| Geometric | 33.32% | 11.34% |
| Uniform | 11.62% | 11.82% |

Table 4: Commit 100 - Successful attacks out of 10000 rounds

| Distribution | Geometric | Uniform |
|---|---|---|
| Geometric | 0.97% | 1.06% |
| Uniform | 0.97% | 1.08% |

With the results we collected we have shown that our theory is indeed correct. The results turned out to be exactly as we anticipated, and therefore can conclude that the attack can only succeed as we discussed previously.

## 5 Related work

While researching solutions to the Two Generals' Paradox, one solution was particularly mentioned all over the Internet [3][2] , and had seemed to solve the problem with a more advanced solution. This solution is what Bitcoin uses in the block-chain, and is known as proof-of-work[2]. In this section we take a look at how the block-chain uses proof-of-work in a rather simplified manner in order to understand the basic working principles and why this does not apply to our situation.

For the proof-of-work, we start by saying a group of five generals want to crack a password hash, but this hash can only be cracked when all five of them collaborate together. If all of the generals do not start at the same time then they will fail to crack the hash and will not obtain the password. This protocol will allow all five generals to know what the time they must start the hash collision.

The protocol starts off by allowing any of the generals who wish to pick a start time to broadcast the network. This can be a single general or all of them. After the broadcast of the timestamps happens, all of the generals wait to receive the times. When a general received a timestamp, the generals keeps that time and discards any others that received. Now that the general has a timestamp, the general will start a proof-of-work including the timestamp in the proof-of-work. The proof-of-work that is being computed here is a hash collision. The general will work to compute a hash, for example, that may contain $x$ 0's at the beginning of the hash. This computation will not be trivial for the general and can take up to ten minutes. All of the generals are computing their proof-of-work and once completed, broadcast the solution to the network.

All of the generals will see the solution broad casted and the timestamp that is associated. If the proof-of-work the general is working on has the same timestamp then the general would take the proof of work and would change the current proof-of-work to include the one that was broad casted which allows for the block-chain to increase in length. If the general is working on a proof-of-work that is not the same timestamp, then the general will switch to the broad casted one because the chain is longer. This process continues on for a couple hours until the length of the block-chain is sufficient.

All of the generals can verify the difficulty of the block-chain and by doing so can see how much parallel CPU power was expended. If the CPU power is great enough to crack the hash then they know everyone must have seen the timestamp. This is because it must have required the majority of the generals to compute a block-chain of that length, meaning those generals saw the timestamp. The generals can now all start cracking the password hash at the timestamp used in the block-chain.

We tried to come up with a solution to the Two Generals' Paradox using this protocol, but we were unable to do so. An important item to note is this protocol is a solution to the more generalized version of the Two Generals' Paradox, known as the Byzantine Generals' Problem. There is a crucial difference between the Two Generals' Paradox and the Byzantine Generals', and that is the network. With Bitcoin, where the block-chain is used money is at stake and if someone tampers with the network then someone is going to lose money. The Byzantine Generals' Problem focuses on fault tolerant environment, which is why the proof-of-work is able to succeed. In our scenario, we may have an attacker that is on-path and both generals need to agree on some item. In the block-chain, the assumption is there is no on-path attacker and the generals don't need to agree, but rather acknowledge that everyone knows the information.

## 6 Conclusion

We have looked into the notoriously difficult Two Generals' Paradox to find possible solutions. We examined the generic fault tolerant scenario where this is no attacker and only a non-reliable network connection. The solution for this scenario, to achieve a 99.999% success rate is to send more than 17 messages.

After examining the fault tolerant scenario, we looked into the way the block-chain implements their solution. Although, it is a very clever implementation, we found that there were a few major differences in the setup of the environments. The block-chain environment does not have on-path attackers and also does not work with agreement, but rather knowing everyone saw the message. Our environment assumed there could be an on-path attacker, even though we have integrity and confidentiality of the information, and both parties need to make an agreement. With these major differences we were unable to find a way to implement the proof-of-work, so we developed our own method.

Our approach to solve the Two Generals' Paradox uses an encrypted and authenticated network channel, but makes the attackers job of fooling the Bank and ATM more difficult. The only way for the attacker to be successful is to drop the last packet in transmission, which means the attacker needs to guess how many commit messages the ATM and Bank are going to transmit for every communication message sent across the network. If the attacker does not guess the correct message number to drop the ATM and Bank will detect the attacker. Even though we do manage to detect an attacker the majority of the time, there is significant overhead with our approach because we are sending many extra messages. All of our code and results are available on our Github repository [4].

## References

[1] E. A. Akkoyunlu, K. Ekanadham, and R. V. Huber. Some constraints and tradeoffs in the design of network communications. In *Proceedings of the Fifth ACM Symposium on Operating Systems Principles*, SOSP '75, pages 67–74, New York, NY, USA, 1975. ACM.

[2] J. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. *Advances in Cryptology - EUROCRYPT 2015 Lecture Notes in Computer Science*, page 281310, 2015.

[3] A. Miller and J. J. LaViola. Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin.

[4] C. West and G. Essertel. https://github.com/wh1t3fox/cs528project.

## A Work Distribution

We both contributed equal parts to this project.