DLC Lab Session 3

In this session, we will deepen our understanding of blockchain through game theory. Now, let's play a game!

1. Prisoner's Dilemma

A *prisoner's dilemma* is a game where two prisoners act on their respective self-interest, but end up receiving a worse overall outcome. Consider two prisoners Alice and Bob. Authorities now try to determine their prison sentences. Alice and Bob are therefore questioned in two *separate* interrogation rooms.

If Alice confesses her crime, i.e., betrays Bob, while Bob remains silent, Bob will receive a 10-year prison sentence, Alice will be released (and vice versa).

If both Alice and Bob confess, both receive a 5-year prison sentence.

If both Alice and Bob remain silent, as the authorities cannot find evidence, both receive a 1-year prison sentence.

Now, let (10,0) represent the situation where Alice gets 10 years in prison, while bob gets 0 years. Similarly, (5,5) represents the situation where both Alice and Bob get 5 years. We can therefore use the following table to describe Alice and Bob's strategies and their respective outcomes.

Table 1: Prisoner's Dilemma Outcomes

Bob → Alice ↓	Confess	Remain Slient
Confess	(5,5)	(0,10)
Remain Silent	(10,0)	(1,1)

Understandably, given a potential outcome (a,b), Alice would aim to minimize her own prison sentence, a. Bob would aim to minimize b. To achieve this, Alice has to choose either to confess or remain silent based on her *belief* of what Bob's strategy might be.

Let's consider Bob's potential strategies, i.e., the two columns. If Bob chooses to confess (the first column), he will get either 5 years or 0 years in prison. Alternatively, if he decides to stay silent (the second column), he will get 10 years or 1 year. Obviously, a rational Bob will choose to confess, as he will be better off with shorter prison sentences.

Given the belief that Bob will confess, Alice now needs to consider her own strategy. If Alice chooses to remain silent, she will get 10 years in prison. If she confesses, however, she will only get 5 years. Therefore, Alice will confess along with Bob, and our end outcome would be (5,5), which is also the *Nash equilibrium* of this game. A Nash equilibrium means that we have now reached a situation where no rational player will change his/her strategy

to be better off. Clearly, this equilibrium doesn't seem optimal to us, as the best overall solution seems to be (1,1), where both Alice and Bob are only jailed for 1 year.

Through the process above, Alice has speculated her opponent, Bob's strategy. She decided that Bob's *dominant strategy* would be to confess, and, according to this *belief*, optimized her own strategy. When we play as Bob and speculate Alice's strategy, will the outcome be the same?

Discuss with your partners why we ended up with (5,5) instead of (1,1)? Is there a mechanism that ensures Alice and Bob will both stay silent?

1.1 Game: Basic Prisoner's Dilemma, 5 minutes

1. If your birthday is between January and June, you will remain silent, if your birthday is between July and December, you will confess. Write down your role (S for silent, C for confess) on your card.
2. As you play this game you should move randomly, i.e., random walk, and change partners each time period.
3. When you encounter a partner, hold your card so that it is visible to you and not to your partner, count '1, 2, 3' together and then simultaneously reveal your cards.
4. Write down your payoff on your score sheet, which can be your card or your phone.
5. Change a partner and repeat.
6. Calculate your average score based on the rounds you've played. Compare your score with others. If your role is C, is your score lower than the S players?

1.2 Game: Prisoner's Dilemma with Communication, 10 minutes

1. You can now choose your role and also talk with each other. However, after each round you should change a partner.
2. If you have a pen, point it upwards for confession. Point it downwards for remaining silent. If you don't have a pen use your finger or figure out an alternative symbol system with your partner. Count '1, 2, 3' together and then simultaneously reveal the roles you choose.
3. Write down your role, your partner's role and your payoff on your score sheet.
4. Change a partner and repeat.
5. Calculate your average score. In general, are you better off with playing Silent or Confession? Discuss with others.

2. Miner's Dilemma

In the last two sessions, we have mined bitcoin locally using Bitcoin Core. Locally, as the mining difficulty is minimal, your machine should be able to generate blocks (50 coins per block) continuously. On the bitcoin network however, mining difficulty is much higher, so individual miners can mine for weeks even months without finding a new block, which is an all-or-nothing situation that reduces the *time value of money*. For example, you are now given two options. One, receive 50 bitcoins after 50 days. Two, receive 1 bitcoin per day in the next 50 days. Obviously, option two is better as you receive some money in advance.

Therefore, most miners will join *mining pools* to reduce the variance of their reward and receive continuous income.

In a mining pool, miners join force in mining, so they mine much faster. When they find a block, they share its reward. A miner's reward is determined by their contribution to the entire pool's computing power, or *hashrate*. For example, a pool has two miners Alice and Bob. Their hashrates are 4 and 6 respectively. When the pool mines 10 bitcoins, Alice will get 4 bitcoins and Bob will get 6, based on their respective shares of total hashrate (40% and 60%).

Specifically, Alice and Bob compute *partial proof-of-works* (PPoWs) and submit them to the pool. The mining pool estimates Alice and Bob's contribution through the rate they submit PPoWs. When Alice successfully find a block, she submits the final solution, i.e. *full proof-of-works* (FPoWs) to the pool and publishes a new block.

However, Alice can choose not to submit FPoWs to the pool. Whenever she finds a block, she chooses to hide it, so it seems to the pool that Alice is a hardworking miner with extreme bad luck. This is called a *block withholding* (BWH) attack.

Let's use the previous example to explain. When Alice becomes a malicious BWH attacker while Bob remains honest, the pool's total hashrate will reduce from 10 to 6 and can only mine 6 bitcoins. However, as Alice still submits PPoWs to the pool, the mining pool recognizes her as a hashrate=4 miner, which means that she will still get 40% of the pool's total mining reward (6 bitcoins). In this case, the pool can only mine 6 coins, in which 2.4 coins go to Alice and 3.6 coins go to Bob.

The BWH attack can be used by mining pools to sabotage each other. A pool can send a malicious "Alice" to infiltrate another pool to undermine its mining efficiency. Let's play a game to see how it works.

2.1 Game: the War of Mining Pools, 40 minutes

1. In this game you will play as a pool miner. Find at least 3 other people to play.
2. For simplicity, your hashrate will be an integer. Use the following terminal command on Linux to generate a random integer between 1 to 10. This number is your hashrate, i.e., how powerful you are as a professional miner, which is also how heavily you will be weighed in your mining pool. Write down this number on paper or simply use your phone's note app.

```
> echo $((1+RANDOM%10)
```

3. Form two "mining pools" in your group. Each pool should have at least 2 people. In each round, both mining pools will be rewarded a number of coins equal to their total hashrate.
4. In the first round, every miner is honest. Calculate how much your mining pool earned (let's call it $r1$) and compare it to the other pool's mining reward r_2. Calculate r_1/r_2 and remember to write it down. This is your pool's relative strength against the other pool. Your goal in this game would be to increase this number.
5. In the second round, some miners become infiltrators. If your Day of Birth (the DD in DD/MM/YY) is an odd number, you will be an infiltrator this round. An infiltrator is an "Alice" who pretends to work for the opponent pool. They do not mine for the victim pool, but they share the reward of the victim pool and bring back the reward to the pool

they belong. Calculate r_1/r_2. If it increases, you win and everyone in your pool wins. If you have an infiltrator on your team and the other pool doesn't, will your pool become relatively stronger? How about your absolute reward r_1?
6. In the next few rounds, anyone can be an infiltrator. Just like in prisoner's dilemma, you can choose your own strategy on whether to infiltrate the other pool. You can discuss with your pool members who should be the infiltrator. Write down your decision on a card and only reveal it to the other pool before you calculate r_1/r_2. You win if this number becomes greater than the r_1/r_2 you calculated in the first round. Try to optimize your pool's strategy. You can also be a silent infiltrator that none of your pool member knows.
7. Next, you can assign yourself a new hashrate between 1 and 30. Play a few more rounds. What effect does the new hashrate have on this game? In what ways do higher or lower hashrates affect your pool's strategy?
8. BWH attack is hypothetically possible, like prisoner's dilemma. However, it does not appear often in real world. What makes it so rare? Discuss with your group members. Refer to Kwon et al. (2017).
9. In this game, a precondition is that mining reward equals to a pool's hashrate. This is true if the two pools are relatively insignificant compared to the entire bitcoin mining workforce. Does the precondition still holds true if the two pools are big enough to affect the global hashrate? What if the two pools are the only mining pools in the world? How will you calculate their mining rewards if they infiltrate each other?
10. Can you think of other kinds of BWH attack? Instead of not telling anyone, what if Alice decides to publish the withheld block eventually? What can she do to profit from this? Refer to Song et al. (2019) for more applications of game theory in blockchain.

3. Let's Mine

We know that bitcoin uses hash functions for block creation. A hash function is a one-way function that takes in data and generates a unique string. There are many hash functions you can use through Linux terminal. Type this in your terminal to use a hash function called MD5.

```
> echo "hello world" | md5sum
```

The result is the hash output of string "hello world". Hash results are unique. For example,

```
> echo "helllo world" | md5sum
```

generates a vastly different result. This ensures that when someone sees a hash output, they won't be able to deduce the original message (there are several ways to crack hash, e.g., using a *rainbow table*). Hash can be seen as an aggressive *lossy compression* method.

Now let's see how hash function is used in bitcoin mining. The following instructions and hash numbers are written by Gimenez (2016).

3.1 The data

Assume this is the hash of the latest block

```
00000000000001adf44c7d69767585
```

To mine the next block, we need to include latest transactions in the network. Here are two simplified transactions that we should add to our new block.

```
5572eca4dd4
db7d0c0b845
```

Mining reward is written in a *coinbase transaction*. It transfers the current block reward to yourself.

```
916d849af76
```

3.2 Building the next block

Let's use a gross approximation of what a new block might look like (the real one uses binary format). It contains the hash of the previous block and the hashes of those 3 transactions:

```
00000000000001adf44c7d69767585--5572eca4dd4-db7d0c0b845-916d849af76--
```

Considering the previous hash, it seems that 13 zeroes is the current difficulty. Our goal is to complete this block with a *nonce* (an arbitrary number) such that the hash of the new block starts with 13 zeros.

3.3 Now let's do mining by hand!

Let's try with nonce=1, and compute the hash of the block:

```
> echo "00000000000001adf44c7d69767585--
        5572eca4dd4-db7d0c0b845-916d849af76--1" | md5sum
8b9b994dcf57f8f90194d82e234b72ac
```

No luck, the hash does not start with a 0. Let's try with nonce=2

```
> echo "00000000000001adf44c7d69767585--
        5572eca4dd4-db7d0c0b845-916d849af76--2" | md5sum
```

No luck. If we pursue until nonce=16, we get our first leading zero.

```
> echo "00000000000001adf44c7d69767585--
        5572eca4dd4-db7d0c0b845-916d849af76--16" | md5sum
```

For nonce=208, we get two leading zeroes!

```
> echo "00000000000001adf44c7d69767585--
        5572eca4dd4-db7d0c0b845-916d849af76--208" | md5sum
```

Continue like this. If you finally find a hash that has 13 leading zeroes, you're a winner! Other miners will now build upon your block.

Now, if there's still some time left, you can discuss with your partners what do you want to do for the group project!

References

Stéphane Gimenez. What are bitcoin miners really solving? - Bitcoin Stack Exchange, 2016. URL `https://bitcoin.stackexchange.com/questions/8031/what-are-bitcoin-miners-really-solving/8034`.

Yujin Kwon, Dohyun Kim, Yunmok Son, Eugene Vasserman, and Yongdae Kim. Be Selfish and Avoid Dilemmas: Fork after Withholding (FAW) attacks on bitcoin. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 195–209, 2017. ISSN 15437221. doi: 10.1145/3133956.3134019.

Li Hua Song, Tao Li, and Yi Lei Wang. Applications of game theory in blockchain. *Journal of Cryptologic Research*, 6(1):100–111, 2019. ISSN 20957025. doi: 10.13868/j.cnki.jcr.000287.