Epicoin

Fundamentally Uncentralizable Cookies:

AL-NABELSI Abd-Al-Kareem,
BALAKHNICHEV Elisey,
OUESLATI Mehdi,
SRIDHAR-KAMAKSHI Ashwin

May 2019

Abstract

**Epicoin is a block-chain system that is designed to help solve NP-Complete problems.
This final project report follows up that of the April, and intends to sum up all of work the Fundamentally Uncentralizable Cookies team achieved during from January through May.**

You can find additional information here: http://www.epitacoin.ml
You can also find the code of the project here: https://github.com/Fundamentally-Uncentralizable-Cookies/Epicoin

Contents

1 Introduction

1.1 What is Epicoin?

Epicoin is a block-chain system that is designed to help solve NP–Complete problems. It is named as the contraction of the names "EPITA" and "Bitcoin".

A block-chain is a chain of blocks linked together by hashes targeting the previous block. A block can be seen as a box that contains its hash and the hash of the previous block, as well as some data that depends on what the block-chain is used for. In the case of Epicoin, this chain of blocks is called "EFOBE", which is an acronym for "Epicoin's Free and Open Blockchain of Epicness".

A hash is the output of a hash function. A hash function is a function that can be used to map data of arbitrary size to data of a fixed size [MMK74]. Hashing is usually a one way street. It is easy to get the hash of an input, but difficult to find the input given its output by a hash function.

What is special about EFOBE is that its blocks are not validated through the state-of-the-art proof-of-work[1] that involves brute-force hashing until a non-trivial number of zero bits start the hash-string. Instead, each miner solves an instance of an NP-Complete problem to make its block valid. As a result, EFOBE can be used as a rainbow table by anyone who is facing an NP-Complete problem.

NP-Complete problems are a class of problems to which no polynomial time answer is known, meaning that solving them for non-trivial cases consumes a considerable amount of resources. This kind of problems is encountered on a daily basis, especially in academia and research. For instance, the following tasks require to solve NP-Complete problems: genome sequencing [MPG12], telescopes' path choosing [AEBCJC06], rocket science logistics [KC10]... An NP-Complete problem shares some properties with reverse hashing: no polynomial time algorithm is known to solve them, but if a solution is found, then the validity of that solution can be checked in polynomial time.

[1] a proof-of-work is a complicated problem that, when solved, proves that its solver had to spend time and resources in doing so, and that that said-solver is indeed the one who solved it.

1.2 What is the problem with the state-of-the art proof-of-work?

One of the main critics toward Bitcoin was induced by its proof of work: because the proof-of-work needs by definition a large quantity of invested efforts, it consumes a large quantity of energy. It is estimated that the Bitcoin block-chain consumed 61.4 TWh in 2018, which a little less than 15% of France's national electricity consumption. [She18])[2].

1.3 How can Epicoin face this problem?

Although Epicoin does not directly tackle the issue of energy consumption, the idea behind Epicoin is to turn one of the main critics against block-chain technologies into an advantage by making use of the massive computational resources needed for their execution: rather than finding a way to make Epicoin's proof-of-work consume less energy, our team decided to reallocate the spent energy on solving another problem that also consumes a lot of energy. This way, the same amount of energy would still be used on each proof-of-work, but the yield of the process would be greater. A new Epicoin coin would no longer be the sole purpose of the proof-of-work's computations, but rather their by-product.

1.4 What is Epicoin's use cases?

NP-Complete problems are faced on a daily basis in both academia and industry. These problems occur in a wide range of areas such as asset management, storage optimization, or production of printed circuit boards...

Both of these actors would benefit from having a list of problems and their solutions as it would be quicker to explore the list than to solve the problem themselves. A layer of abstraction could even be added on top of Epicoin in order to further facilitate the reading process by having a server observe the Epicoin network and keep a sorted list of Epicoin's solutions.

From a financial perspective, the value of Epicoin should be backed by the fact that these actors would benefit from keeping the EFOBE alive. Having a minimal threshold for its value, a better and optimized version of Epicoin[3] could to a certain extend guarantee that its mining is profitable given that the offer does not exceed the demand too much. Fortunately, if there is more offer, the table grows bigger and interests more people. Thus, we have at least in theory a guarantee that a minimal number of miners would be present at any time on the Epicoin network.

[2] Out of intellectual integrity, we must point out that some consider Bitcoin's electricity consumption to be acceptable [Kat18]

[3] We consider that our S2 project is more of a proof of concept than a real scalable solution.

2 Problem Analysis and Decomposition

One of the first things we had to do was to break down the big problem that building Epicoin is into smaller sub problems.

> "Epicoin is a **block-chain** system that is designed to help **solve NP-Complete problems**."

2.1 Epicoin is a block-chain system.

This means that our software must produce a block-chain. A block-chain is a chain of blocks linked together by hashes targeting the previous block. A block can be seen as a box that contains its hash and the hash of the previous block, as well as some data that depends on what the block-chain is used for.

Hence, our software has to generate an immutable stack such as each element of the stack is the defined by the previous. This implies that we have to write a **validator** that would check that each new block respects the condition to be on top of the previous block.

Furthermore, it also implies that we need a **Block Generator** that would be able to create new blocks respecting the later conditions.

2.2 Epicoin needs a hashing algorithm.

A hash is the output of a hash function. A hash function is a function that can be used to map data of arbitrary size to data of a fixed size [MMK74]. Hashing is usually a one way street. It is easy to get the hash of an input, but difficult to find the input given its output by a hash function.

We need the hash functions to be able to uniquely indentify each block, and thus use that identifier to make the chain of blocks.

2.3 Epicoin is designed to help solve NP-Complete problems.

The peculiarity of Epicoin is that its proof-of-work solves NP-Complete problems. This implies that we have to solve **NP-Complete problems** in order to fulfill the conditions needed to generate a new block. The **Block Generator** thus becomes the **Solver**, and its purpose is to solve an NP-Complete problem defined by the block which precedes it.

2.4 Epicoin must be decentralized.

Obviously, Epicoin must be a decentralized system. A decentralized system is when the system runs without having any central server in charge of keeping track of its state. Instead, all actors of the system are connected to each other and form a peer-to-peer network, meaning that they all act as both a client and a server.

If Epicoin was not decentralized, it would allow the central server to bend the rules of the blockchain by, for instance, refusing to transmit the solutions found by some miners.

Each instance of the Epicoin software thus has to communicate with a network of other instances of this same software.

2.5 Epicoin's components have to work in parallel

We have so far deduced that the Epicoin system needs a Checker, a Solver, and some solution to communicate with other nodes of the network. We can thus notice that all of these components have to work in parallel for the system to be efficient: if the solver pauses the networking actor, there is no way for the solver to know when someone else has issued a new block, and all of the extra computations would go in loss. Likewise, if the checker doesn't work while the solver or the network actor does, there is no way for the solver to know if the claim received by the network actor is true or not.

2.6 The solver has to be as efficient as possible

The system only works if the miners get the best times out of their equipment. If we want to approach these minimal times, the solver has to be as efficient as possible. This is why we wanted to make use of the Graphical Processing Units of the machine as well.

3 Prior Research

3.1 AL-NABELSI Abd-Al-Kareem

Since he is less-technologically fluent than the rest of the team, Abd-Al-Kareem's research was more focused on learning on topics related to the project rather than directly facing problems encountered.

He first crammed the Bitcoin paper[Nak] and later studied on Application Layers[Exp], Transmission Control/ Internet[EGL], and User Datagram[POS80] protocols (TCP, IP, and UDP).

Once that was done, his studies were oriented toward enhancing his technical skills: he studied on compression algorithms (mainly RLE[Tea], LZ77[Com], and Huffman coding[HS]).

To make sure that he understood well the materials he studied, and in order to be productive during this first phase, Abd-Al-Kareem wrote articles about each topic. These articles are published online on our website: www.epitacoin.ml#portfolio.

Abd-Al-Kareem kept writing these articles later on, adding new ones at each period.

3.2 BALAKHNICHEV Elisey

Elisey's research work was primarily focused on the "Miner logic" sub-module, and on utility-related problems.

Elisey's work on "Miner logic" includes research on preventing block forgery and on deterministic random number generation. The details of his research work can be found in articles published on our website: www.epitacoin.ml#portfolio.

Elisey's work on block forgery allowed us to identify a major problem in our original idea, which was that the usage of NP-Complete problems instead of hashing algorithms for the proof-of-work was inefficient at best, and flawed at worst.

3.3 OUESLATI Mehdi

Mehdi's research work was oriented toward identifying and solving the problems related to the "Peer-to-peer communication" sub-module, which he was responsible of.

Mehdi studied how various decentralized solutions were designed. The list of the systems he studied is: Freenet[CSWH], GNUtella[Tri], Kademlia[BM], The Onion Router[Pis05], the BitTorrent protocol[KK], and, obviously, Bitcoin[Nak].

Mehdi's research work is the origin of the networking solutions we implemented. These solutions can indeed be summed up as a compilation of the solutions he cherry-picked to face the problems induced by our project.

The networking problems will be detailed section 3.1. The solutions chosen to face them will be detailed in section 4.1.

3.4 SRIDHAR-KAMAKSHI Ashwin

Ashwin's research work was mainly about the "Miner logic" sub–module, which he was responsible of.

Ashwin researched on "Consensus algorithms" and on concepts such as the "Byzantine Fault". Unfortunately, because of the Forgery problem discovered by Elisey, the solutions that Ashwin had proposed were abandoned. His research work on the subject can still be found as an article on our website: (www.epitacoin.ml#portfolio).

Ashwin also researched on block-chain structures, and his work led to the current EFOBE design described in section 4.2.

4 Solution Details

In this section, you can find more information about how we tackled some of the main problems we faced.

4.1 For peer-to-peer communication

The peer-to-peer communication sub-module has the purpose of connecting the different Epicoin users to each others so they could then apply their miner logic and interact with each others.

4.1.1 How to join the network...

4.1.1.1 ...when there is a DNS seeder

The first node in the network will be our own, and we will include its IP address in the downloading files so that new-comers can easily join the network by pinging it. This node will act as a **DNS seeder**, meaning that it will send to the new-comer the information needed for it to connect to its neighbours.

This approach is the one taken by Bitcoin, which uses an IRC server to welcome new-comers.

4.1.1.2 ...when there is no DNS seeder

It is still possible that the DNS seeder is down or compromised. If that happens, the new-comer will not be able to join the network using it.

In such scenario, two cases are to be taken into account: either the new-comer knows nodes that are connected to the network, or it does not.

If it does, which may happen if it was already a part of the network before disconnecting from it, it will try to reach them and use them as DNS seeders.

If it does not, which is unfortunately the by default case, it will send UDP requests to random IP addresses hoping that one may answer its need.

4.1.2 How to keep the network alive

To face the isolation problem described in section 3.1.2, we are implementing a Key-Based Routing system[BM]. This means that each node will be associated to a unique identifier, and that the closer two identifiers are, the more likely it will be that the nodes know each other.

Knowing that, nodes will continuously try to connect to new nodes by asking their neighbours if they know the node that is identified by a given key. As they are thus continuously making new connections, the rate of creation of new links should be greater than that of disconnection of nodes, and the network should thus survive.

4.1.3 How to protect from Man In The Middle attacks

Internet is, as its name suggests, a network of interconnected machines. This means that packets most often go through several machines before reaching their destination. One of these machines could be malicious, and either sniff [4] the traffic or intercept the packet and send a forged one instead. The former action would represent a danger to the anonymity of the users of the network, and the latter could compromise the proper functioning of the network.

To protect from Man In The Middle attacks, we are going to use Secure Shell **(SSH)**. Basically this means that each packet sent by a node will include the following data:

- **Packet Length**, which is encoded on 4 bytes and which indicates the length of the packet.
- **Amount of Padding**, which is encoded on a byte, and which indicates how much padding there is in the packet.
- **the Payload**, which is the actually useful data sent
- **Padding**, which is useless random data sent so that a potential sniffer will not be able to get the real size of the message.
- **the Message Identification Code**, which is here to guarantee that there was no forgery.

All the data, except the Packet Length of the Message Identification Code are then encrypted with the destination's node public key.

[4] i.e. watch and log

4.2 For miner logic

The miner logic's sub-module has the purpose of checking the validity of the EFOBE and generating its new blocks.

4.2.1 Insuring block validity

The Block architecture of EFOBE consists of two inherent classes. The Local Validity and the Global (tree) Validity. Given the nature of our Blockchain, The Block will have a set of parameters to facilitate its validity, namely:

- **The Problem:** This serves as the identifier for which problem the block is currently solving for.
- **Parameter for the problem:** These are the general parameters from which the problem is solved upon.
- **Solutions:** This houses the completed solution of the problem.This part is redundant for a unsolved block as it won't exist in the network.

The Local validity of a block is determined by a state of reversal.Basically it says that if a problem has been solved using the said parameters provided,then the reversed solution must be valid to the given problem itself. So the block must accurately implement the Solver and Validator Classes which in turn is the basis for the Miners Logic in EFOBE.

For the Global Validity, we consider the EFOBE to be a form of tree where each block Parent has a set of children and so on.The root of this tree will be static.
Now to prove the validity of a block which has (now) completed the computations for a problem and has a solution,The Blockchain takes the parameters of the block itself and the parameters of the previous block to create a link to the Parent in the EFOBE.

So the local and global validities in conjecture with each other provide the necessary stability required to mitigate threats posed by potential hackers and forgers .

4.2.2 Chaining solutions to problems

When a problem has been solved by a block and a solution has been produced, the next step is to link the block to its respective parent. This is achieved by first taking the core hash of the block and comparing it with its corresponding Parent Hash by using a HashMap. After this is executed, the block is now successfully connected to a parent node.

4.2.3 Concurrent EFOBE Trees

As of now, there is no hard solution which mitigates this effectively. A good alternative would be to keep the concurrent trees until a high enough threshold has been reached. This would make the smaller branch negligible in comparison which gives the Blockchain fair grounds to omit it from the main tree

4.3 Security vs Performance

4.3.1 Pretty Good Compromise

While debriefing from our first presentation, we took into account the critics that were made toward us. We were overly ambitious the first time, and although we believe in the usefulness of our project, we forgot that as students our main purpose was to present a concrete working prototype to a jury. We thus had to make our program work in a short time regardless of how the actual crypto-currencies do. One of the implications of this was that we decided that we could not use RSA to encrypt everything. We therefore decided to reduce our security to make our program faster, and we implemented "Pretty Good Privacy" (PGP)[cgi].

Pretty Good Privacy is actually a complex program with many features. The two main ones that we needed were that of Confidentiality and of Authentication.

4.3.2 Confidentiality

The PGP approach for the Confidentiality feature is the following:

1. Randomly generate a 128 bits symmetric key. That key will only be used one unique time only.
2. Encrypt that key using the receiver's public RSA keys.
3. Encrypt the to-be-sent message using that key

Then do everything in reverse for the decryption.

In practice, we used AES[20116] encryption, our own RSA[Mil09], and the System.Cryptography.RNGCryptoServiceProvider for the random generator.

4.3.3 Authentification

The PGP approach to the Authentication is the following:

1. hash the message
2. encrypt the hash using the sender's private key
3. concatenate the encrypted hash to the message

Then, the receiver goes through the following steps:

1. decrypt the hash using the sender's public keys
2. hash the message using the same hash function
3. compare the hashed message to the decrypted hash to know if the sender was indeed the right one

In practice we used our implementation of SHA256 and that of RSA. We plan on replacing SHA256.

4.4 Concerning Hashing

4.4.1 SHA256

We had already started writing SHA256 previously, but it did not pass the unit tests. We spent a lot of time understanding the way it worked and debugging it, but it allowed us to better understand the theory behind it.

4.4.2 Generalization: Keccak-C

As we were later concerned about using other hashing functions, we went generic by rewriting "Keccak-C"[LPGR]. "Keccak-C" is a sponge[5] function. This means that it is a generic function for both stream ciphers and hashing functions. It can take a word input of arbitrary size and output another word of a given size. In "Keccak-C", the blocs of the input are xored with initial bits and then permuted with each others.

"SHA" hashing functions derive from the "Keccak-C" one as they are that same function with given constants. For instance, "SHA256" is a "Keccak-C" with the constants (512;1600).

[5] For the trivia, we called our Keccak-C function "Bob".

4.5 Scalability

After being introduced to how C# handles multi-threading, we decided to pay homage to our programming classes and to use different tasks to make our program run in a parallel style.

4.5.1 Multithreading[NJ]: division of tasks

We took a look back on our previous work and decided to make a new division of asynchronous tasks. The parameters we took into account when doing so were what resources each task needed and on what other tasks each depended. We then tweeked our code so each task would run on a parallel thread.

4.5.2 Loading and binding NP-Complete problems

Long term general solution involves multi-layer abstraction - with solver on the top and problem instance on the bottom, multiple wrappers-binders laying in the middle would allow easy transition on either end of the layers. The solution towards which currently converges the development involves creating a problem wrapper class, topologically directly below solver, which wraps around the problem interface representation, transforming between problem's type parameterization (of parameters and solution) and internal representation (for transmission). Next layer - problem interface, is, well, an interface implemented by targeted actual problems implementation, requiring implementation of type parameterized solving and checking methods,: in case of DLL-based, the parameterized interface can be directly implemented by classes inside external DLLs.

Actually loading and binding the interfaces to the wrapper layer involves locating all target DLLs, binding them dynamically through Assembly, iterating over declared classes assignable to the public interface, and instantiating found problem implementations through reflection. Later on, when and if, the switch to OpenCL will be performed, only the interface-implementation layer will have to be modified - instead of being implemented by API users in other DLLs, implemented by whatever OpenCL interface there will be.

4.5.3 Multi-threading: Inter-thread communication

Following the work we reported in the previous project report, we went in depth into Inter Thread Communication. The purpose of Inter Thread Communication[Int] is to orchestrate the behaviour of threads. For each thread, we made an "inbox" queue of messages. Then, when a thread is active, it would have a specific time to read the messages it received at its own pace. That specific time is decided for each thread depending on what task it is supposed to achieve and how it works.

Since each thread reads its messages at least once per cycle, the communication achieves asynchronicity. Moreover, since data needs to be encapsulated in a message to be passed to other threads, there is no conflict of resources. We hence chose to favour economy in computing power over economy in memory space. The reason for that is that we will test our code with small inputs on a laptop when we will be in front of a jury. We thus preferred it to run quickly so the demo does not look boring.

4.5.4 Multi-threading: its relevance

The multi-threading dimension of our project is crucial for its use. On the networking side, we need both the Baby and the Parent class to work at the same time to keep the network up and running. Similarly, we need the miner logic to communicate with the networking components without interruption because any delay in the mining process could virtually cost money to the user.

4.6 For additional problems

4.6.1 Ensuring code quality

4.6.1.1 Unit tests

To make sure that our code works properly, we wrote unit tests. Unit tests are basically methods that will input data of which the output is known to the tested piece of code, and check that its output is correct by comparing it to the already-known results.

4.6.1.2 Documentation

We document a fair part of code we write after finishing a given story. Documentation can be either an external ".pdf" or ".pptx" file, or a wall of comments. It is preferably both.

5 Methodology

5.1 For the first period

In order to ensure work efficiency and well accordance in task divisions, the Fundamentally Uncentralizable Cookies agreed on a three--phase work methodology.

5.2 [27/01 → 15/02] Research Phase

During the first phase, the team members made in-depth research about the issues they would have to solve before the next presentation. They also met once a week to sum up the current state of their research-work and explain to the team how relevant their findings were to the project.

You can find more about the research in section 3.

5.3 [22/02 → 02/03] Implementation Phase

During the implementation phase, the weekly meetings stopped and we communicated through social medias or on occasional meet-ups at school. Everyone's priority was to implement the stories they were assigned to.

To ensure that progress was being made, Abd-Al-Kareem was responsible of making sure that Ashwin, Elisey and Mehdi were regular on their work; and Ashwin was responsible of Abd-Al-Kareem.

5.4 [08/03 → 09/03] Group Debriefing and Interview Prep-up

As deadline approached, we had to get ready for the presentation. We thus met up so each person could pitch his work to the others, and we merged all of our offline work.

This phase could seem trivial, but it was so full of git merging errors, bugs, and Object-Oriented re-designs that it had to be mentioned. We learned an important lesson, and we will never **not progressively** "*git commit git push*" again.

5.5 For the second period

In order to ensure work efficiency and well accordance in task divisions, the Fundamentally Uncentralizable Cookies agreed on a two-phase work methodology. This is a change from the previous time were we had to dedicate a whole period of research. The reason of this change is that the research we previously did already covered most of the theory we had to put in practice this time, and that we noticed that when putting in practice we still had to do extra research.

5.5.1 [16/03 → 14/04] Implementation Phase

During the implementation phase, we met once a week to check on the work's progression. The rest of the time, we communicated through social medias. Everyone's priority was to implement the stories they were assigned to.

To ensure that progress was being made, Abd-Al-Kareem was once more responsible of making sure that Ashwin, Elisey and Mehdi were regular on their work; and Ashwin was again responsible of Abd-Al-Kareem.

5.5.2 [12/04 → 21/04] Group Debriefing and Interview Prep-up

As deadline approached, we had to get ready for the presentation. We thus met up so each person could pitch his work to the others, and we merged all of our offline work.

This time, we learnt the lessons from last presentation. We put more efforts into making our work presentable and went by the motto of "Show, don't tell". For instance, we not only added printing messages for the application to interact with the console, but we even made it so everything would be clear. Later, we worked on a Graphical User Interface before having our Application Programming Interface yet.

5.6 For the third period

After the success of our second presentation, we decided to keep the same methodology. Especially that this time we barely had a thing to do. The only things left were rather short tasks, and what we really had to do was merge all of our code together, and handle the bugs and issues that came from that.

5.6.1 [28/04 → 14/04] Implementation Phase

During the implementation phase, we met once a week to check on the work's progression. The rest of the time, we communicated through social medias. Everyone's priority was to implement the stories they were assigned to.

To ensure that progress was being made, Abd-Al-Kareem was once again responsible of making sure that Ashwin, Elisey and Mehdi were regular on their work; and Ashwin was again responsible of Abd-Al-Kareem.

5.6.2 [15/05 → 04/06] Group Debriefing and Interview Prep-up

Our main focus is now to prepare our presentation. The main challenge is that we were mostly focused on the technology so far, and we now have to come back at the idea and try to think from the consumer's perspective to appeal.

5.7 Projet Management Methodology

The SCRUM methodology was chosen to execute this project. The global idea was thus decomposed into modules and sub-modules. Each sub-module will be later decomposed into stories, each representing one precise functionality to implement, or one precise task to achieve. These stories will then sequentially move from one state to another following the progression of the project. The states are "to be ", "emergency", "work in progress", "to be tested", "done".

Each module was assigned to one supervisor who has the responsibility of making sure that that module is advancing at an acceptable pace. This does not mean that the supervisor has to take care of all the tasks in the module he is responsible of, but rather that he will have a privileged voice on the matter, and that he will have to keep track of the progress made in that module. Ideally, every member will do a little bit of everything while keeping a clear and reliable perspective of the current state of one aspect of the project.

The list of modules was proposed, argued, and voted for. It represents what the team collectively thinks of as being the most logical division of tasks. The modules were later assigned weights that estimate how long the task should take to be done. To compute the weights, we used the following formula:

For

M_i the estimated percentage of total time needed to complete the ith module,

$E_j(x)$ the estimation of the value of x by the jth team member,

$S_k^{M_i}$ the estimated percentage of M_i needed to complete the kth sub-module of the ith module,

$S_{(k,i)}$ the estimated percentage of total time needed to complete the kth sub-module of the ith module,

$$\begin{aligned} M_i &= (\sum_{j=1}^{4} E_j(M_i)) \times \tfrac{1}{4} \\ S_{(k,i)} &= (\sum_{j=1}^{4} E_j(S_k^{M_i})) \times \tfrac{M_i}{4} \end{aligned}$$

Once the weight of each module was computed, the modules were then distributed in between the team members through vote, while making sure that each member was assigned supervision on an estimated 25% of the total work, by a maximum discrepancy of 2%.

During the discussion about task division, differences of experience were noticed, which led some more technical modules to be under the supervision of some members rather than others. However, it was made clear that this was only a matter of organization, and an emphasis was put on the importance of the implication of each member in every stage of production.

5.8 Sharing Responsibilities: Module Distribution

Here follows the list of main modules and sub-modules with the name of the main [6] as it was originally submitted. of each[7]:

Mehdi OUESLATI **Distributed Networking**

- 19.43% **Peer-to-peer communication** Mehdi OUESLATI
- 12.95% **Request handling** Ashwin SRIDHAR-KAMAKSHI

Ashwin SRIDHAR-KAMAKSHI **Project Management**

- 2.78% **Deadline checks** Abd-Al-Kareem AL-NABELSI
- 0.35% **Group cohesion** Abd-Al-Kareem AL-NABELSI
- 1.27% **Meeting organization** Abd-Al-Kareem AL-NABELSI
- 3% **Communication with teachers** Mehdi OUESLATI

Abd-Al-Kareem AL-NABELSI **Communication**

- 4.17% **Reports** Abd-Al-Kareem AL-NABELSI
- 4.17% **Website** Abd-Al-Kareem AL-NABELSI
- 7.06% **Documentation** Abd-Al-Kareem AL-NABELSI

Abd-Al-Kareem AL-NABELSI **User Interface**

- 3.08% **Application Programming Interface** Abd-Al-Kareem AL-NABELSI

Elisey BALAKHNICHEV **Inner Software Architecture**

- 8.80% **Miner logic** Ashwin SRIDHAR-KAMAKSHI
- 7.96% **OpenCL reader/binding** Elisey BALAKHNICHEV
- 16.77% **Hashing** Elisey BALAKHNICHEV

Elisey BALAKHNICHEV **NP-Complete Algorithms Implementation**

- 2.74% **Travelling Salesman Problem solver and checker** Ashwin SRIDHAR-KAMAKSHI
- 2.74% **Knapsack Problem solver and checker** Elisey BALAKHNICHEV
- 2.74% **Clique Problem solver and checker** Mehdi OUESLATI

Please keep in mind that this module distribution is just an indicator to express who is in charge of keeping track of each module's progress. The list of achieved modules can be found in section 6.

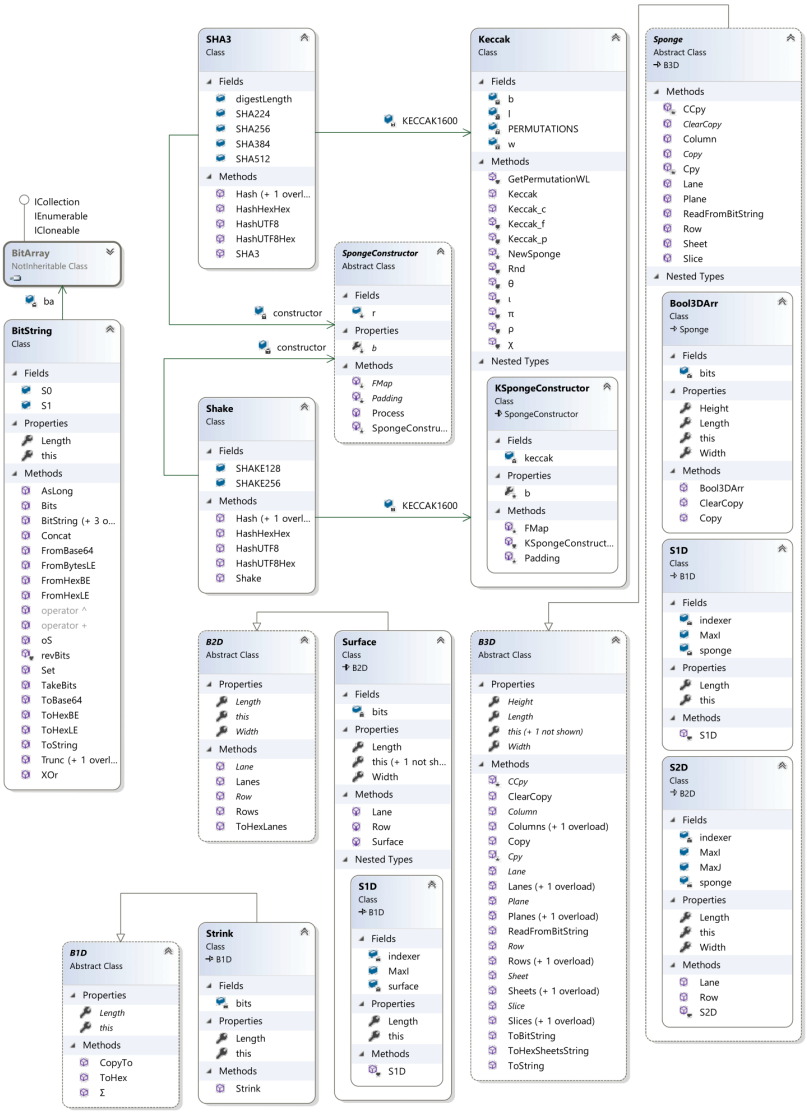
[6] For each module and sub-module, the substitute of Abd-Al-Kareem is Elisey, the substitute of Elisey is Mehdi, the substitute of Mehdi is Ashwin, and Ashwin's substitute is Abd-Al-Kareem

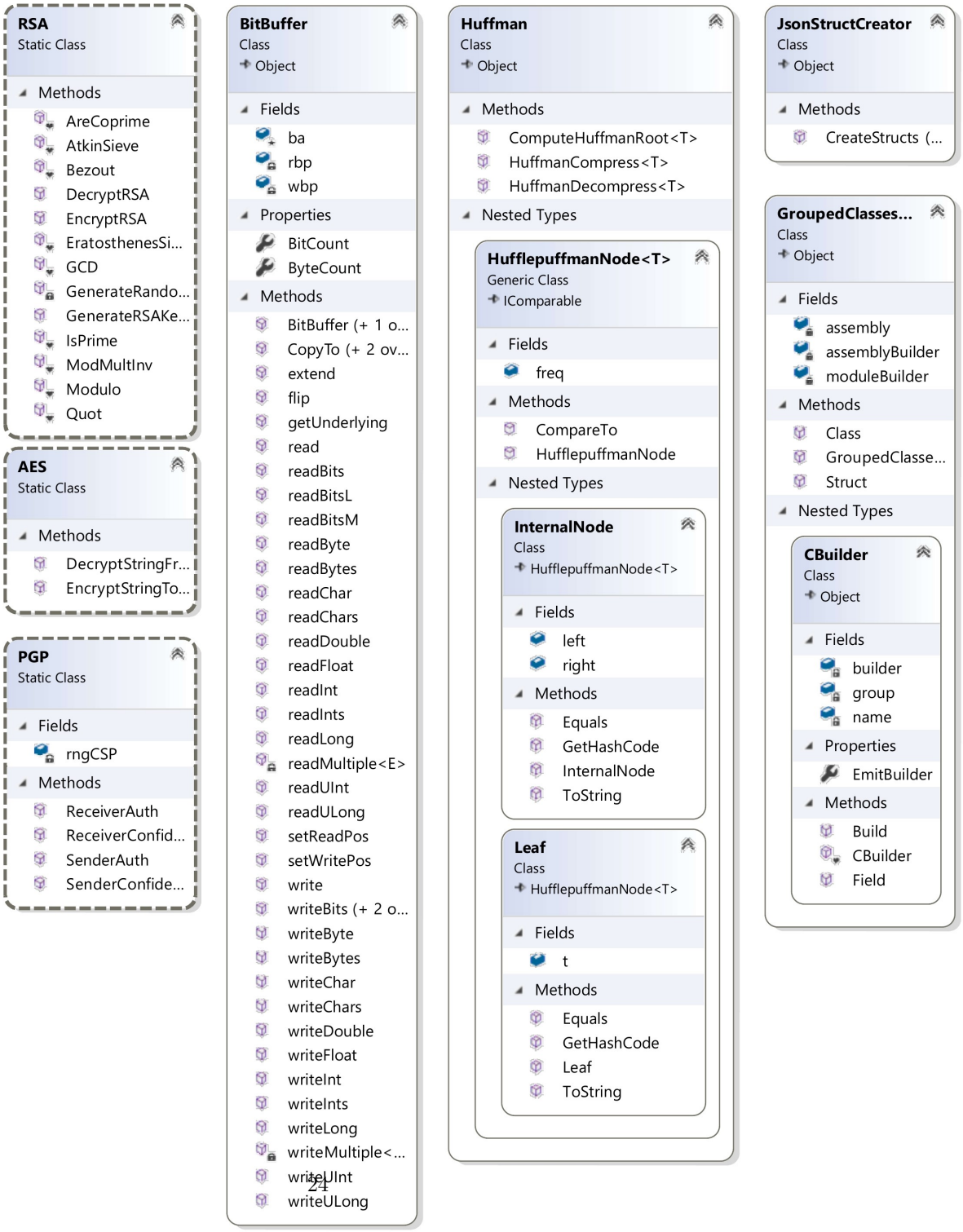
[7] Expressed percentages are estimated percentages of total working time

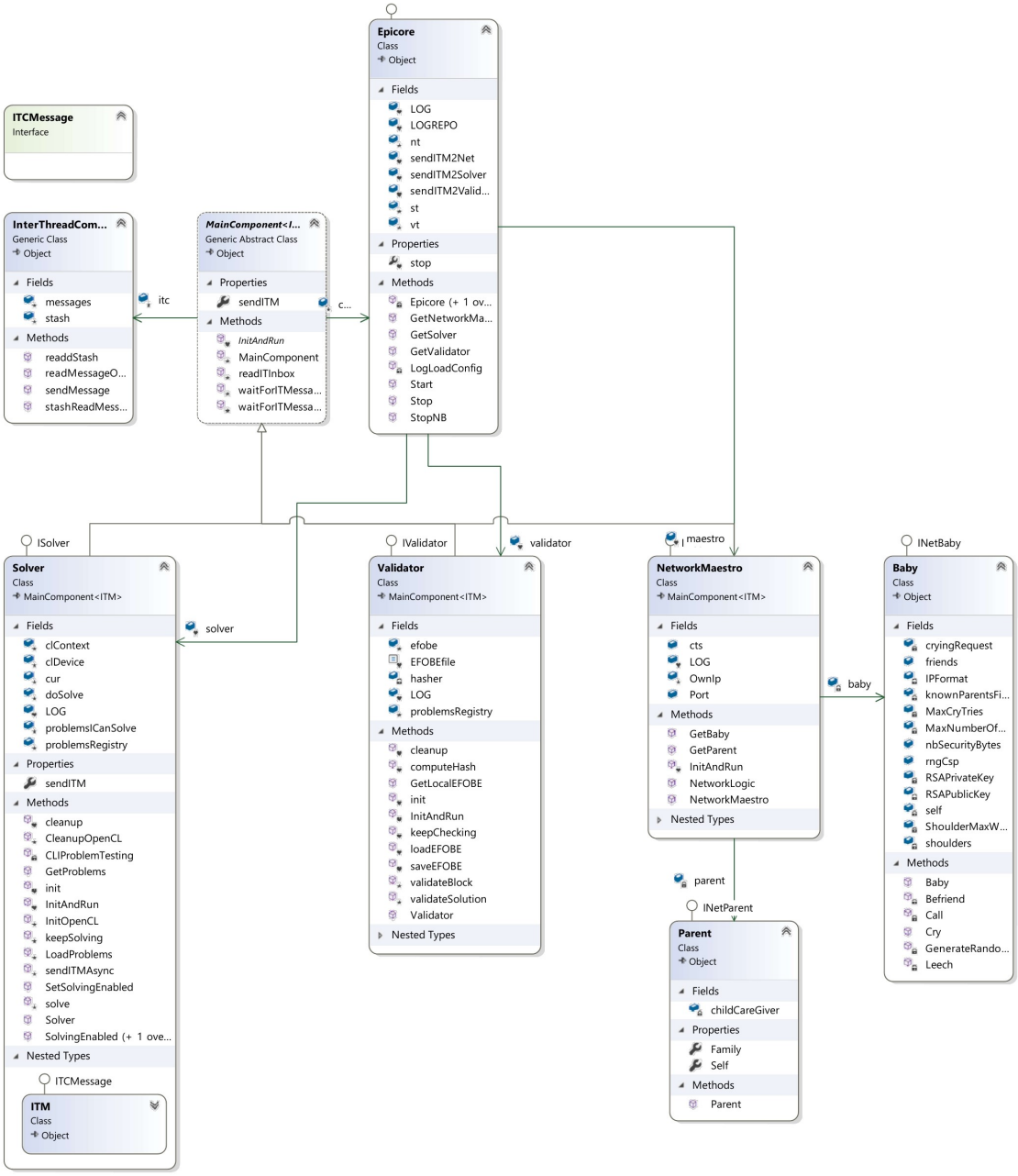
6 Achieved Work

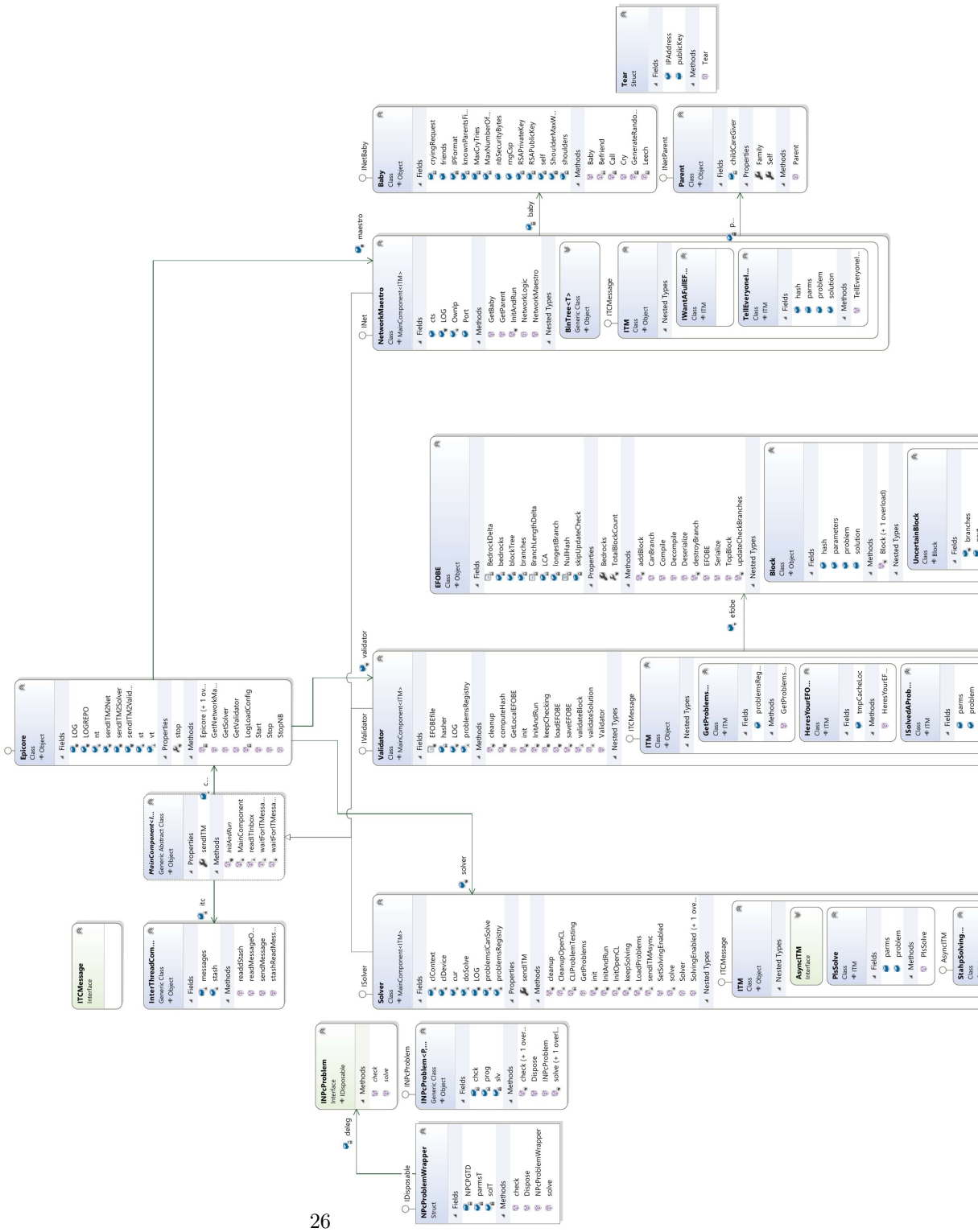
6.1 Unified Modeling Language Diagram of Epicoin

Here follows the UML diagram of our project, displaying our software architecture and summing up all of the code we wrote:




RSA
Static Class
Methods
AreCoprime
AtkinSieve
Bezout
DecryptRSA
EncryptRSA
EratosthenesSi...
GCD
GenerateRando...
GenerateRSAKe...
IsPrime
ModMultInv
Modulo
Quot
AES
Static Class
Methods
DecryptStringFr...
EncryptStringTo...
PGP
Static Class
Fields
rngCSP
Methods
ReceiverAuth
ReceiverConfid...
SenderAuth
SenderConfide...
BitBuffer
Class
Object
Fields
ba
rbp
wbp
Properties
BitCount
ByteCount
Methods
BitBuffer (+ 1 o...
CopyTo (+ 2 ov...
extend
flip
getUnderlying
read
readBits
readBitsL
readBitsM
readByte
readBytes
readChar
readChars
readDouble
readFloat
readInt
readInts
readLong
readMultiple<E>
readUInt
readULong
setReadPos
setWritePos
write
writeBits (+ 2 o...
writeByte
writeBytes
writeChar
writeChars
writeDouble
writeFloat
writeInt
writeInts
writeLong
writeMultiple<...
writeUInt
writeULong
Huffman
Class
Object
Methods
ComputeHuffmanRoot<T>
HuffmanCompress<T>
HuffmanDecompress<T>
Nested Types
HufflepuffmanNode<T>
Generic Class
IComparable
Fields
freq
Methods
CompareTo
HufflepuffmanNode
Nested Types
InternalNode
Class
HufflepuffmanNode<T>
Fields
left
right
Methods
Equals
GetHashCode
InternalNode
ToString
Leaf
Class
HufflepuffmanNode<T>
Fields
t
Methods
Equals
GetHashCode
Leaf
ToString
JsonStructCreator
Class
Object
Methods
CreateStructs (...
GroupedClasses...
Class
Object
Fields
assembly
assemblyBuilder
moduleBuilder
Methods
Class
GroupedClasse...
Struct
Nested Types
CBuilder
Class
Object
Fields
builder
group
name
Properties
EmitBuilder
Methods
Build
CBuilder
Field


ITCMessage
Interface
InterThreadCom...
Generic Class
Object
Fields
messages
stash
Methods
readdStash
readMessageO...
sendMessage
stashReadMess...
itc
MainComponent<I...
Generic Abstract Class
Object
Properties
sendITM
Methods
InitAndRun
MainComponent
readITInbox
waitForITMessa...
waitForITMessa...
c...
Epicore
Class
Object
Fields
LOG
LOGREPO
nt
sendITM2Net
sendITM2Solver
sendITM2Valid...
st
vt
Properties
stop
Methods
Epicore (+ 1 ov...
GetNetworkMa...
GetSolver
GetValidator
LogLoadConfig
Start
Stop
StopNB
ISolver
Solver
Class
MainComponent<ITM>
Fields
clContext
clDevice
cur
doSolve
LOG
problemsICanSolve
problemsRegistry
Properties
sendITM
Methods
cleanup
CleanupOpenCL
CLIProblemTesting
GetProblems
init
InitAndRun
InitOpenCL
keepSolving
LoadProblems
sendITMAsync
SetSolvingEnabled
solve
Solver
SolvingEnabled (+ 1 ove...
Nested Types
ITCMessage
ITM
Class
Object
solver
IValidator
Validator
Class
MainComponent<ITM>
Fields
efobe
EFOBEfile
hasher
LOG
problemsRegistry
Methods
cleanup
computeHash
GetLocalEFOBE
init
InitAndRun
keepChecking
loadEFOBE
saveEFOBE
validateBlock
validateSolution
Validator
Nested Types
validator
maestro
NetworkMaestro
Class
MainComponent<ITM>
Fields
cts
LOG
OwnIp
Port
Methods
GetBaby
GetParent
InitAndRun
NetworkLogic
NetworkMaestro
Nested Types
baby
parent
INetParent
Parent
Class
Object
Fields
childCareGiver
Properties
Family
Self
Methods
Parent
INetBaby
Baby
Class
Object
Fields
cryingRequest
friends
IPFormat
knownParentsFi...
MaxCryTries
MaxNumberOf...
nbSecurityBytes
rngCsp
RSAPrivateKey
RSAPublicKey
self
ShoulderMaxW...
shoulders
Methods
Baby
Befriend
Call
Cry
GenerateRando...
Leech



6.2 Achieved work by deadlines

6.2.1 By the 11th of March

6.2.1.1 List of completed networking stories

OUESLATI Mehdi *Epinet/Baby.cs* < **Baby**>**.Call(**<**Friend**>**)**, tries to open a web socket connection with a Friend. If the friend answers the call, it is added to the Parent class's family list. Otherwise, it is deleted from the friend list.

OUESLATI Mehdi *Epinet/Baby.cs* <**Baby**>**.Cry()**, sends a UDP request to a random IP asking it if it can get Baby in the network.

OUESLATI Mehdi *Epinet/Baby.cs* **Implement RSA**, RSA is one of the most efficient asymetric existing cryptic algorithms. It will be useful for the implementation of SSH and for the <Baby>.Cry() method. The current implementation uses Eratosthenes' sieve to generate prime numbers, but this will be replaced by Atkin's, which is more efficient.

OUESLATI Mehdi *Epinet/Baby.cs* **Friend class**, a Friend instance is a node known by the Baby class. Its most defining properties are an IP address and a KBR one.

OUESLATI Mehdi *Epinet/Baby.cs* **Overall Structure Design**, designed the baby class with its fields and methods in order for it to do one precise task which is to connect the node to the network and ensures that it stays connected.

OUESLATI Mehdi *Epinet/Baby.cs* **Potential Parent structure** the Potential Parent struct is useful for the <Baby>.Cry() method. A Potential Parent is a machine to which Baby sent a tear.

OUESLATI Mehdi *Epinet/NetworkMaestro.cs* **NetworkMaestro.cs**, this class supervises the Parent and Baby class. It also contains some constants that the two classes have in common, like the port or the cancellation token source. The main purpose of this class is to serve as a gateway between the networking logic, **Epinet**, and the miner logic, **Epicore**.

OUESLATI Mehdi *Epinet/Parent.cs* **Overall Structure Design**, designed the parent class with its basic fields and methods. Its main purpose is to handle request, and will be further developed before the next deadline accordingly to what was specified in the specification paper.

OUESLATI Mehdi *Epinet/Parent.cs* **Tear structure**, this struct is for when a parent receives a baby's UDP request. Its most defining properties are an IP address and a public key.

6.2.1.2 List of completed miner logic stories

AL-NABELSI Abd-Al-Kareem *EFOBE.cs* **EFOBE**, which stands for *Epicoin's Free and Open Blockchain of Epicness*, is so important that we implemented an entire dedicated class for its management.

SRIDHAR-KAMAKSHI Ashwin *NPcP.cs* **Solver**, we needed a single class managing all the problems, delegating and interpreting calls to problems, from internal API. Though it is directly linked with binding loading problems (which is in progress), the delegation is done and ready.

SRIDHAR-KAMAKSHI Ashwin *NPcPTest.cs* **Solver Unit Testing**, making sure that Solver interprets generics correctly and no exceptions arise in any possible direct interaction between solver and some implemented problem.

AL-NABELSI Abd-Al-Kareem *EFOBE.cs* **Validator**, the validator validates EFOBE blocks on both local & global levels. It is one of the main Epicore components, created and fully implemented to ensure integral structure of EFOBE and not allow cheaters in.

AL-NABELSI Abd-Al-Kareem *EFOBETest.cs* **Validates the Validator**, it is important to thoroughly test validity of your validator, so it does not output invalid false-positives. One may wonder... should something validates the validator validator..?

6.2.1.3 List of completed utility stories

AL-NABELSI Abd-Al-Kareem *Epinet/Util.cs* **Compression**, Elisey and Kareem chose Huffman Coding as a compression algorithm for data exchange and EFOBE compression. Abd-Al-Kareem wrote the initial binary tree class and the unit tests for Huffman coding.

BALAKHNICHEV Elisey *Epinet/Util.cs* **Compression**, Kareem and Elisey chose Huffman Coding as a compression algorithm for data exchange and EFOBE compression. Elisey implemented the actual Huffman Coding algorithm.

BALAKHNICHEV Elisey *https://github.com/Fundamentally-Uncentralizable-Cookies* **GitHub Coordination**, all of our stable code is on Github under M.I.T beerware licence. Elisey took care of dealing with merge issues and making sure that no major issue happened on git. His help was extremely valuable during the Git-chaos of the third phase. Lessons were learned then.

SRIDHAR-KAMAKSHI Ashwin *Epinet/SHA256.cs* **Implementing SHA256**, Ashwin wrote the SHA256 hashing algorithm. That is useful to give unique identifiers to blocks or nodes.

BALAKHNICHEV Elisey *Core.Tests/UtilsTest.cs* **Implementing SHA256 Unit Tests**, Elisey wrote the unit tests for SHA256. Our current version of SHA256 still does not passes all the tests, but it is anyway due for the next deadline.

SRIDHAR-KAMAKSHI Ashwin *Epicore.cs* **Main()**, everything needs to be tied together, executable and managed from single place, unrelated to either components; especially with multithreading.
Therefore, every main component <Solver>, <Validator>, < NetworkMaestro> is a child of <MainComponent<ITM>>, and is instantiated and managed by Epicore, which also manages threading. Default client bootstrapping is done through the Main in Program.cs.

AL-NABELSI Abd-Al-Kareem *www. epitacoin. ml* **Website** Kareem wrote the website's HTML structure and filled it with content. He was notably in charge of writing articles. [8]

SRIDHAR-KAMAKSHI Ashwin *www. epitacoin. ml* **Website** Ashwin was in charge of the website's design, which he did in CSS. He also added animations using Javascript, and handled the adaptativity of its layout.

6.2.1.4 List of completed additional stories

BALAKHNICHEV Elisey *Epicore.cs* **Inter Thread Communication**, now that everything runs on a dedicated thread, we can't simply invoke a method in Solver from Network. We need to establish concurrency-safe communications system between threads. Based on message-queues model, the ITC is currently in its early development stages, and is expected, still ahead of time, for second presentation.

BALAKHNICHEV Elisey *Epicore.cs* **Multi-threading** When Epicoin is solving a problem, we still need for it to be responsive to any incoming updates (for example EFOBE updates). Therefore every main component of Epicoin must run on a separate thread not to block one another.

BALAKHNICHEV Elisey *NPcP.cs* **Problem Loading Binding** By definition, Epicoin should be able to solve any problem, given that it is NP-Complete. Therefore we need to be able to ship problems (in whatever form) apart from core code. Overall solution involves multi-layer abstraction through wrappers and reflection, in order to load solutions through dynamic DLL binding at first, OpenCL compilation in the end, and seamlessly switch during development in-between. In progress, dll-binding expected for second presentation.

[8] By the way, we strongly encourage you to go on the website to see additional content such as the result of our researches.

6.2.2 By the 23rd of April

6.2.2.1 List of completed Pretty Good Privacy stories

OUESLATI Mehdi *PGP.cs*, **SenderAuth** is a static method that allows a sender to generate a message which proves to the receiver that the sender is indeed the person that they pretend to be.

OUESLATI Mehdi *PGP.cs*, **ReceiverAuth** is a static method that allows the receiver to verify, given a message that was supposedly generated using SenderAuth, that the sender is indeed the person that they pretend to be.

OUESLATI Mehdi *PGP.cs*, **SenderConfidentiality** is a static method that allows a sender to encrypt the message that they want to send faster than by using RSA. The trick is to use AES as a complement. As for now, our version of AES does not pass the unit tests, so the version we will present uses System.Cryptography.AES for the symmetric encryption.

OUESLATI Mehdi *PGP.cs*, **ReceiverConfidentiality** is a static method that allows a sender to decrypt a message that was encrypted using SenderConfidentiality.

6.2.2.2 List of completed Front-End stories

AL-NABELSI Abd-Al-Kareem *Epicore, Epicore.cs¿GUI¡ITM¿*, **Graphing System Performance**. As we needed some form of analytics for the user's disposal,We have implemented a simple Performance Monitor using Windows Forms.Additionally all the data is graphed in real time using the corresponding elements in Visual Studio.

AL-NABELSI Abd-Al-Kareem *Epicore, Epicore.cs¿GUI¡ITM¿* **Integration of the Performance Monitor with the Main GUI**. To integrate the functionalities of the performance monitor to the main app, A new class was made to allow for multiple pages to exist in a single WinForm giving the application multi page functionality.

AL-NABELSI Abd-Al-Kareem *Epicore, Epicore.cs¿GUI¡ITM¿* GUI Landing, **Landing Page/ Page of interaction**. Citing the need for a GUI interface, A landing page had to be made to provide a simple and easy to use interface for the end-user.The theme of this GUI is subtly based around our website which bears similar attributes.

AL-NABELSI Abd-Al-Kareem *GUI Animations* **Landing Animations**. The addition of a background animation to the landing page makes the page look more appealing to a potential end user. Its objective is to remove the monotonous nature of a generic WinForm .

SRIDHAR-KAMAKSHI Ashwin *WebView¡ITM¿*, **CoreWebView**. For the WebView visualisation, A tree structure protocol had to be put into place to ensure the incoming data can be visualised accordingly.This facilitates for a more accurate representation of EFOBE.

SRIDHAR-KAMAKSHI Ashwin *WebViewjITM¿*, **MiscWebView**. In addition to the Core part of the WebView, A legend has been created along with labelling for each major node with a short popup of information which contains the nature of the block,its problem and its ongoing solution.

6.2.2.3 List of completed Hashing stories

BALAKHNICHEV Elisey *Utils.cs*, **Custom Bit Storage** C BitArray is nice, but actually not. We wrapped around it to create an actually useable immutable BitString
.NET already has a class for storing and manipulating bits individually in an efficient way. It is however lacking in methods, is easily mutable, and specific to runtime system bit-endianness. So, because Keccak functions must be system independent, sponge construction functions specify bit endianness standard to use for input interpretation. In result, we created a custom [publicly] immutable bit-storage structure compatible with Keccak standard and providing all necessary methods for easy implementation.

BALAKHNICHEV Elisey *BitDim.cs* **Multidimensional bit storage**, Keccak uses three dimensional bit structure - state array, and sub-dimensional views for its' round operations
We operate in OOP and therefore why not, for educational purposes, represent Keccak operations in their raw 3D definition? For that, we need equivalent classes representing bits in one, two and 3 dimensions, as well as sub-dimensional views (i.e. modifying bit 7 on lane 1,3 modifies the original bit in the 3D state array as well [in this case, 1,3,7]).

BALAKHNICHEV Elisey *Keccak.cs* **Keccak** θ, Keccak θ sponge operation (XOR each bit in the state with the parities of two columns in the array)

SRIDHAR-KAMAKSHI Ashwin *Keccak.cs* **Keccak** ρ, sponge operation (rotate the bits of each lane by a length, called the offset, which depends on the fixed x and y coordinates of the lane; equivalently, for each bit in the lane, the z coordinate is modified by adding the offset).

OUESLATI Mehdi *Keccak.cs* **Keccak** π, Keccak π sponge operation (rearrange the positions of the lanes).

AL-NABELSI Abd-Al-Kareem *Keccak.cs* **Keccak** χ, Keccak ι sponge operation (modify some of the bits of lane 0,0 in a manner that depends on the round index [the other 24 lanes are not affected by ι])

OUESLATI Mehdi *Keccak.cs* **Keccak** χ, Keccak π sponge operation (rearrange the positions of the lanes).

OUESLATI Mehdi *Keccak.cs* **Keccak round**, A single round (iteration) of composed keccak sponge operations

OUESLATI Mehdi *Keccak.cs* **Keccak-p**, The great Keccak permutation from which all hashing functions are constructed. Consists of some consecutive number of round applications.

SRIDHAR-KAMAKSHI Ashwin *Keccak.cs* **Keccak-f**, Specific family of Keccak-p permutations. Consists of parametrization of Keccak-p.

SRIDHAR-KAMAKSHI Ashwin *Keccak.cs* **Keccak Sponge Constructor**, Sponge construction function constructing [fixed-size] sponge from [varying length/streaming] bitwise input, through two phases - absorbing and squishing.

SRIDHAR-KAMAKSHI Ashwin *Keccak.cs* **Keccak[c]**,Sponge construction used with Keccak-f permutations and 10*1 padding rule.

BALAKHNICHEV Elisey *Keccak.cs* **SHA3**,bitwise hashing functions, specific variations of Keccak[c], applying and identifying input to the suffix. Standard SHA3 variations: 224, 256, 384, 512.

BALAKHNICHEV Elisey *Keccak.cs* **SHA3 XOF (SHAKE)**, SHA3 extendable/variable output function variations, also known as Shake (an identifying suffix different from SHA3 is used).

6.2.2.4 List of completed OpenCL stories

BALAKHNICHEV Elisey *Json2POCO.cs* **Runtime struct builder**, Create/generate new classes at runtime using Emit and IL.
We will need to upload parameters (and solutions for checking) to the Open CL device. Usually simply creating an equivalent(ly aligned) structure in the cl script and c is sufficient, as Marshal allows byte-wise copying between devices (automatically respecting bit endianness). In our case however, the input data format is dynamic (at least to us as the Epicoin creators - those writing problems must define exact structs, cl scripts and data format), that is the cl script can be anything and underlying Problem and Solution structs as well. In the meantime, we receive actual input data instance from network in JSON format. One way to upload it could be to manually parse JSON and try to correspond each entry with one in the cl script (which gets exponentially harder with number of embedded structs)... or we could generate said equivalent structures at runtime following a joint JSON definition.

BALAKHNICHEV Elisey *NPcP.cs* **Re-wrap interface layers from DLL binding to Open CL**, Modify bottom layers of I problems structure, adapting to Open CL
We foresaw a mutation of bottom-level implementation of problem solving and validation, and therefore created a layered structure for DLL binding. Meaning that now, when we want to switch bottom-layer to Open CL, we only need to modify, well, said bottom layer (and all top-level accesses are left untouched!). This part is still on progress, though.

BALAKHNICHEV Elisey *NPcP.cs* **Open CL binding**, We need to first bind devices and load Open CL in general, before we can compile individual kernels and upload problem information (as it relies on general pre-binding). This part is still on progress, though.

6.2.3 By the 17th of May

6.2.3.1 Application Programming Interface

AL-NABELSI Abd-Al-Kareem *Core/EPI.cs* **Epicore Gateway**, Provides methods for starting/stopping the core, as well as accessing components; aka all methods that any self-respecting epicore [implementation] must implement.

AL-NABELSI Abd-Al-Kareem *Core/EPI.cs* **Epicore Solver**, Solver component of the Core. Responsible for loading problems (on initialization) and solving them when given the parameters, if solving problems was enabled.

AL-NABELSI Abd-Al-Kareem *Core/EPI.cs* **Epicore Validator**, Validator component of the Core. Responsible for validating problems solutions (both by-self and received) as well as the EFOBE.

AL-NABELSI Abd-Al-Kareem *Core/EPI.cs* **Epicore INet**, Networking component of the Core. Mainly calls to the Maestro class.

6.2.3.2 Request handling

SRIDHAR-KAMAKSHI Ashwin *Epinet/Parent.cs* **Tear handler**, Receives babies' tears and answers to them if they are valid.

SRIDHAR-KAMAKSHI Ashwin *Epinet/Parent.cs* **EFOBE Block download**, Downloads the blocks it misses if its neighbours have a longer EFOBE.

SRIDHAR-KAMAKSHI Ashwin *Epinet/Parent.cs* **EFOBE Block upload**, Uploads the blocks requested by a neighbour.

6.2.3.3 Travelling Salesman Problem

SRIDHAR-KAMAKSHI Ashwin *(external file)* **Travelling Salesman Problem Solver**, a program solving the travelling salesman problem.

SRIDHAR-KAMAKSHI Ashwin *(external file)* **Travelling Salesman Problem Checker**, a program checking that a given attempt at a solution to the travelling salesman problem is valid.

6.2.3.4 Knapsack Problem

BALAKHNICHEV Elisey *(external file)* **Knapsack Problem Solver**, a program solving the knapsack problem.

BALAKHNICHEV Elisey *(external file)* **Knapsack Problem Checker**, a program checking that a given attempt at a solution to the knapsack problem is valid.

6.2.3.5 Clique Problem

OUESLATI Mehdi *(external file)* **Clique Problem Solver**, a program solving the clique problem.

OUESLATI Mehdi *(external file)* **Clique Problem Checker**, a program checking that a given attempt at a solution to the clique is valid.

6.2.3.6 Git-related tasks

BALAKHNICHEV Elisey *https://github.com/Fundamentally-Uncentralizable-Cookies/Epicoin/pulls* **Validating PRs**,Ensuring code format and validity of large components when merged into master. Any large change should be developed on a dedicated branch - one of good git practices. When complete, its' addition to the master should be requested using a pull request, which should be reviewed by the responsible team for respecting style, format, and standards - everything that is not already checked by CI in general.

BALAKHNICHEV Elisey *https://github.com/orgs/Fundamentally-Uncentralizable-Cookies/projects/1 [non public]* **Project board management**,Managing targets for development in the project board. Project board is a very useful tool for managing To-Do, Work in Progress, and completed tasks, and though it partially self-manages thanks to integration with pull requests and issues, someone still needs to add and move around tasks not directly represented on github itself.

BALAKHNICHEV Elisey *Git Repo* **Propagating master**, Propagating changes through the repository on dependent branches. Once a change is accepted into master (or a small one is directly committed), it needs to be propagated onto dependent branches to ensure synchronicity with the main branch everywhere. Generally speaking, the propagation has to be performed on any change on a non-child-most branch onto its' children, but this mostly concerns the master branch.

BALAKHNICHEV Elisey *https://github.com/Fundamentally-Uncentralizable-Cookies/Epicoin/branches/stale* **Cleaning up outdated branches** Deciding which feature developments branched off and archiving old branches. For in-dev/semi-archival purposes, sometimes some features branch-off due to similar-level feature to be developed on top of first. Eventually these branch-offs "replace" the original feature branch, in the sense that all the development is done there, and the first branch serves no more purpose other than archival. Such branches should thus be managed properly - archived, or sometimes deleted.

7 Team's Experience

7.1 AL-NABELSI Abd-Al-Kareem's comment on the experience

Throughout our project i believe it have changed the way i go around doing things and it increased the amount of knowledge that i have in regards to networking and coding significantly ,in the beginning,the idea of doing some sort of cryptocurrency instead of a game was very challenging, at least for me,because i have a huge experience in the field of games, i've been working with games for a long time,so to go against the wave was a breath of fresh air and i knew since we agreed on our project that i have to do an enormous amount of research,because while other colleagues are coming up with new ideas, i had yet to dive into the concept and see how can we execute it,but with the help of our team,i kept on learning new things such as compression algorithms,networking,decrypting and how the whole blockchain system works.

We enjoyed our time together as a group throughout our first presentation and we thought we had something special to offer,I believed that we have a great potential and insisted on doing multiple meetings and practice sessions so that we can get the atmosphere of the presentation going on since I could not offer much in our first presentation for my lack of knowledge, and our team was doing a great job,I loved the spirit that we had thus I knew how to drive the team,I remember we had some issues in the beginning in regards to git,I had known git from Epita and nowhere else and I was very confused of how we're going to go around and start coding "Together" , but thanks to Elisey he taught us how to use it. I was honored to have such a great team with such amount of knowledge that I can ask them about almost anything in regards to computer science and they would sufficiently answer me which made my life 10 times easier,even when I was writing the code for our website[9] the team was very supportive and we were all happy with how it turned out and with a touch from our one and only Ashwin,he have made it a very modern/beautiful looking website that represents and respects our team.

Furthermore, after our first presentation, our team spirit was not the best due to our ability to show something visual,but because the nature of our project, we thought we didn't had much to show, Our website was live and we had made our network live as well but there wasn't any visual aspect to our project besides the website to that point.somehow that went over our head but we couldn't let it hold us down.In addition, We started coming up with different ideas after our first presentation,and we discussed a more efficient ways to go around our project, more visual aspects, showing something that no other cryptocurrency provides,thus i wanted to make the Gui,and after some research i found that almost all cryptocurrencies user need to monitor the usage of their Cpu,Ram and

[9] which i didn't had an experience in HTML5

wattage consumption in order to determine the efficiency of their machines,and almost all GUIs were not showing the wattage consumption which was very critical,and so i was able to implement it in our GUI,Also with the help of Ashwin we had a live visual version of our project in our website so that almost anyone can understand what's going on[10], and i think we've successfully made a friendly/easy to understand interface for everyone to see.

I think being an international team coming from all over the world, from Europe,from Asia and from Africa really helped our creativity and inspired us to spend more time working together, we had more meetings than our previous presentation and we enjoyed it overall, not only talking about project but talking about the differences in our culture and how we go around doing things, and that can be seen in our website and Gui design, it was plenty of fun. Back to coding[11],I had wrote the structure of our block "EFOBE" and some unit test, and before the second presentation, Elisey had done Sha 256 which was due to the second presentation but we've already started it anyway because we're cool like that,but it needed more tweaking as it didn't pass all the unit tests, and we were concerned about using other hashing functions, so we implemented the keccak family and divided them into our group,so when we arrived to our second presentation, we were very proud and confident so we showed what we had and it was very satisfying, our team has made a great job.

I personally have enjoyed working with out team ,i've discovered some different programming habits, different way to make my life easier when i'm coding,and trying to code outside the boundaries/footnoteas in doing something that i wanted because i needed as suppose to giving an objective such as in our programming class and it seems that programming is really like a canvas which you can do whatever you want the way you want to,and it's your own canvas,you always learn about new things and if you think you're stuck,there's a whole punch of friendly people whom are willing to help at Epita.

Furthermore for our last presentation, we're pretty excited about it since it's probably the last time we're going to work on this project[12] and we're looking into creating something different since our project is different from the usual "game project", at this point we've reached the fun part of our project which is the Np complete problems and since we've already started OpenCL,we're feeling very confident once again, our goals for each presentation have successfully fulfilled each time and after we finish implementing our Algorithms, we will have a fully functional cryptocurrency which is yet to be test but will be before the last presentation. We've been working really hard on this project, we've spend countless number of hours on it, you can ask Mehdi[13]The only thing

[10] Yes i know it glorious :D
[11] or as Mehdi says "Back to work"
[12] hopefully not the last project
[13] He loves unit tests :D

that we're missing is the final touches,some bug fixes, design, maybe more user friendly interfaces, but that's all to be determined once we have a working product. And the last aspect that we have to work on/footnotethe team still doesn't know about it YET is the way we're going to present our project, in the beginning we were putting everything together, it was like making a cake,first we got the components, then we mixed them together and we cooked them with and we will have a great cake, but what's a good cake with a bad presentation.

7.2 BALAKHNICHEV Elisey's comment on the experience

All along, working together, as a group of four people was a very nice experience, and so far still is. We have researched together numerous times in various topics, which allowed us to explore much faster the possible solutions. As for working alone, though seemingly nothing in particular at first - simply alone, considering everyone else was also working alone created a peculiar atmosphere of everyone kinda working together but separately, at the same time[14].

Our organization was not the best at first, and though we did have a few aggregational meetings[15] before the first presentation, a significant portion of work had to be done in the last weekend prior to the first presentation. Aggregating everything there, merging gits and other documents was particularly fun. It was interesting to learn from every subgroups' research about different algorithms and compression and thoroughly compare them; just as was reading about exploits and flaws of different crypto currencies.

Speaking of Git, oh did we *git gud* with it. Most of the team wasn't particularly experienced with git in the beginning, so happily i had a few years of experience with me. Though it did not prevent sudden appearance of some problems "out of nowhere"[16] due to poor git maintenance in some forks. In the end, i think learning to git properly paid off very well for the entirety of the team.

Also Git kept us awake for longer. And by that i mean not the Git itself, but rather Mr.Travis the CI that was thoroughly checking our repository everytime we committed yelling personally at us by email, every time something failed. We did believe the code we wrote in Wordpad (for C) and Echo in the terminal[17] (for all other files) would compile every time, but all mighty Mr Travis still managed to prove us wrong, every time.

[14] kinda like multithreading

[15] forced-organized by our meetings mastermind Abd-Al-Karim

[16] yeah, so, i did manage to eventually explain why they appeared much later on, but it still made not much sense

[17] or in some cases powershell on Windows 10

Obviously Mr.Travis integrated flawlessly Unit Tests. I just wouldn't forget about them, would I - the true beauty of modern high level languages, Unit Tests, topping off the 3 horsemen of sleepless nights[18] of the last weekend prior to the first presentation, accompanied by Git and Mr.Travis the CI himself. Though, with the help of Mr.Travis, they did ensure our code, by training us like neural networks - with a satisfying green check mark every passing commit and a menacing red cross every failing, they still had to be written - which is whole other story of fun.

The first presentation took place, and the results weren't bad, but partially disappointing, though matching the work supplied and shown for/on the presentation. So we organized a meeting to correct our mistakes for the next presentation, be more organized with regular weekly meetings from now on[19], and get better with everything in general.

The organization for the second presentation was thus improved thanks to the experience gained from the first, though everything was, once again, aggregated in the last weekend prior to the presentation. By that time Git was working smoothly[20] and together with Mr.Travis allowed fluent and efficient parallelization of work.

The team meanwhile, was still the team of same four people, and relatively speaking, that's already a great thing. Though it doesn't mean everything will get worse here on forward by every word you read - I just wanted to disappoint you with nothing to be sad about[21]. So, everything was great, the team is still great, everyone is having fun, working together is just as entertaining and informative as it was before, if not more - all aspects of teamwork have improved overall.

In the times between the two first presentations, i had first focused on hashing. And as seemingly correct and rather simple hashing functions seemed during implementation, they simply did not work. Correcting them usually did not help, and I did try completely rewriting them multiple times. All while i was writing more and more semi-duplicate code for hashing functions themselves as well as underlying data structures[22], i still simply could not understand why on Earth would the hashing functions not work, despite matching documentation an even multiple other sources.

[18] This is an exaggeration, please don't take seriously - still no one here wants to believe that i slept the 3 nights of the last weekend prior to the first presentation.

[19] though it was not perfectly respected every week, it did improve our workflow significantly

[20] left aside a couple out-of-date local clones

[21] yes, yes, i know, a repeated joke is not as fun eventually. But this is the last time, i promise

[22] actually useable bit-strings mostly, as we wanted bitwise hashing functions for whatever reason

Considering you're reading this close to, or after the final project presentation, you should imagine that we eventually figured it out, and you would be right. The problem turned out to be an issue with bit endianness, and not "standard" endianness, as DOT NET employs standards matching most hashing algorithms on byte level. You see, we decided to implement our hashing functions on *bit* level, and this is where Keccak documentation and C environment differed. Once i confirmed that to be the issue and fixed it, everything just magically worked[23].

This entire debacle with hashing functions ended up making me implement the entire Keccak hashing family in C, in object-oriented data structures, which also turned out surprisingly simple to understand the mechanisms underlying hashing functions[24]. It is so clear and simple, you can even use it to demonstrate Keccak rounds to kids - and they absolutely loved when sponge function Bob came into play!

I can also mention that our implementation is, at this moment, the only DOT NET implementation of Keccak hashing functions operating on bits, as all other existing implementations[25] require byte sized and/or aligned input... All we needed, and all we use from this entire hashing library is the SHA3-256 hashing function.

The hashing being finally complete, it was time to move on.. Forth on to Open CL! Though i should note that at this stage, i mean OpenCL bindings and inter-operation with C code, as OpenCL code itself, that is OpenCL kernels are written in OpenCL C (based on C99), and the actual problems we will need to write will be/have been written later on, that is in time for the third presentation[26].

Creating the bindings themselves for OpenCL API and kernels was not complicated, as C and C DLLs are mostly interoperable, we have however met the true first fun problem when it was time to move problems from DLLs[27] to OpenCL kernels. You see, we allow any problem to be defined, solved and checked, as long as, well, it can be defined, solved, and checked in OpenCL. Which means, as long as it representable in OpenCL, the parameters and solution(s) to the problem can be anything. At the same time, we need to be able to send said parameters and solutions through the internet, as well as consolidate

[23] well, almost - i still had to eliminate 2 typos

[24] originally such readable structure was obviously intended for debugging

[25] in DOT NET

[26] at the time this footnote is being written, they are work in progress, but will be ready for the final presentation

[27] our progressive development ment that problems were first implemented and binded through use of DLLs, and then by modifying some of layers of the abstracted binding structure, would be moved to OpenCL

them in the EFOBE. Considering they can be anything, we needed something very dynamic and abstract.

And now, let me teach you a lesson in trickery - you obviously can, in such situation, use a JSON to store the information, and upload/download it to/from the computing device using Marshal on each bottom field (and calculating memory alignment through kernel source seeking). Or you can use one JSON to define structs to be generated at runtime in C, that are memory-equivalent (field-aligned) with their counterparts defined in OpenCL kernels, and then use another JSON to store the actual data, which you simply serialize/deserialize into generated C structs, which you communicate with the computing device by simply Marshaling the entire structs! I ensure you, this is not an overkill: this method only supports a few types of data - only all the ones supported by OpenCL[28].

It was time for the second presentation. And the results we received did not disappoint, and i mean - they did match the work done for the presentation, so that was great. After all, the only significant comment received during the presentation was concerning the style[29] of that, and previous, presentations, which we planned to adjust for the final one anyway.

The work for the third presentation has started, following the same overall planning, though we have decided to omit a couple meetings due to approaching finals[30], by aggregating content for discussion into the remaining meetings. Team spirit is at its' strongest, and everything is going as great as it always has.

For the final presentation we have set a goal (obviously) - to make Epicoin actually work, in real time, and we're making everything[31] to make it happen. And by that i mean mostly finishing up, polishing up, squishing some bugs here and there... You know, mostly mundane stuff and final touches, to the core itself as well as the user interface.

While the networking part is being handled by others, i am making sure the core is stable, all problems are working and new Lowest-Common-Ancestor based EFOBE model does not go wild in any scenario. I am ensuring safe concurrency and multi-threading of different components, as well as their intercommunication. OpenCL will also be finalized to properly support arrays[32],

[28] char, unsigned char, uchar, short, unsigned short, ushort, int, unsigned int, uint, long, unsigned long, ulong, float, double, and half floating point types

[29] for your information - i am kind of a creative lead for the development of the of slides/background imagery for the presentations

[30] that was not the only reason, nor the primary one, but others shall remain unmentioned in this document

[31] in the realm of plausible and believable - we will not summon daleks to boost us, even if we can

[32] there's a deal where arrays in C in structs remain pointers, due to being able to be variable size

and while i am at it, i will also finish writing and check the actual NP complete problems to be run on the demonstration EFOBE.

Overall, the group has remained very supportive of each-other throughout the entirety of development process. Everyone is rather reactive, except a couple delays[33]. Everyone keeps pace, and despite partitioning of work, and often parallelization across members or sub-groups everyone still manages to stay in sync and more or less know everything that is going on.

Personally, the project has been fun and interesting. Occasionally a kind-of brain-opening experience, while slightly annoying at other times[34]. But it did allow me to *see sharp* through dark magic-y reflective tricks, and write, what i still consider, some wonderful sponge functions, about Bob and Patrick.

Scapegoats...
A good scapegoat is nearly as welcome as a solution to the problem. - *Elisey Balakhnichev*

7.3 OUESLATI Mehdi's comment on the experience

"It's a lot easier to do good work when you have good words to say and work with good people." *-Mark Harmon*

In order to talk about Epicoin, I really need to split the question into separate parts. I indeed feel like there was, obviously, a technical aspect to this project, but also a strong educational dimension, and an even stronger human face.

From a technical perspective, I was in charge of networking, and I was the person behind the Epicoin solution. This made me work both as a theorist and as a network engineer. I really enjoyed working on both sides, and using designing theoretical solutions to abstract problems before implementing them. My job also implied a lot of cryptography which, in turn, implied reading a considerable number of research papers.

I really enjoyed working in the frontier of theory and implementation, and I really had fun matching the dots with the concepts I found. This experience really made me want to apply to the EPITA LSE laboratoy in third year, and maybe even to take the "Systèmes Réseaux et Sécurité" major later on.

[33] all of which took place before the first presentation
[34] *cough* hashing *cough*

On the educational side, as I am a mostly self taught person, I used to be in a hacky "move fast and break things" mindset were I mainly focused on making stuff work and did not really care if I had to start over everything because of some flawed designed choice I made. This time was different. Because I already had to design software solutions before I joined Epita, my fellow team members relied on me and on my opinion about a plethora of technical problems they faced. This meant that I was confronted to tough calls and had to really weight the pros and cons of each decision even more than if it was mine alone because I could not ask my teammates to rewrite everything they had done in case I was wrong. I really believe that this experience taught me how to be more "humble" and "wise" in my approach. This is exactly what I wanted to learn by applying to this school, and I am sure that I still have much more to learn on the subject.

About the human aspect of things, I would start by saying that we are a Russian, an Indian, a Jordanian and a Tunisian who all come from different backgrounds and have different centers of interest. Elisey won a Russian regional championship of robotics, and had already worked with git and C before; Ashwin is an experienced UX and front-end developer and used to do freelance. Kareem is the oldest member of the group and he used to work in an IT department. I was thus confronted to a heterogeneous group of people with different skill sets.

Our first challenge was obviously to find a way to work in harmony. It was really tough at first, and it took us more than a week to decide on our team's name. The reason of that is that we did not want to rule by the majority, but rather by unanimity, and this meant that we had to discuss with others, explain our point of view and, more importantly, listen to what the others have to say.

Fortunately, we quickly found some common points between us, and we mainly focused on them. For instance, it turned out that we all had similar taste in humour[35], which was a good start. I really have to point out to Kareem's contribution in forging our team spirit. He organized meetings at kebab places and made us pull coding all-nighters together.

An interesting phenomenon that the ruling by unanimity induced is that some untold rules appeared within the group. The first one of which is: everyone has a respected authority on their own field. What I mean by that is that we all had a clear role in the team's dynamic and that this fact was obvious to everyone.

Elisey, for instance, was clearly the programming lead. If anyone had troubles implementing something, he would be the one we went to for help. He also was the mastermind behind our coding architecture and the ultimate referee in terms of git issues.

[35]Fun fact: since the creation of the site, one can find a picture of Joseph Stalin singing The Internationale and telling us to go to work on www.epitacoin.ml/rev.html

Kareem was in some way our team manager. We respected his opinion on deadlines, and we let him be in charge of the team cohesion. Moreover, when we had a conflict in task repartitions, which did not happen often, he was the one to settle it. We also trusted him with our problems and he provided comfort. I was in a coding strike when I heard that my grandfather had a stroke *(He is now fine though)*, he is the one who noticed that something was wrong with me and he calmed me down and offered me to rest.

Ashwin was our fixer. He always find ways to get around logistic problems, and he was the one who brought a fresh start to our conversations when we were stuck in a dead end.

Even though we had our ups and downs, I really enjoyed working with my teammates on this project. I may not keep a great interest in block-chain technology, which I now believe is very limited by design; but I will surely remember the fun times I had with those whom I shall call friends, and I am proud of what we achieved together as a team. My only disappointment is that Elisey and I could not find a better solution to the NP-Complete based proof of work than to just use the EFOBE as a rainbow table.

7.4 SRIDHAR-KAMAKSHI Ashwin's comment on the experience

From the beginning to the end of this project, it has been an amazing journey to work with the EPICOIN team.I learned so much on the subject of Blockchain and I truly believe that does knowledge would not just help me now, but for the foreseeable future as well.

Being the core front end developer for this project , I was able to showcase my talents in Design and serve a simple and easy to use interface for a broader audience.I truly believe that apart from all the bells and whistles that a product may provide, the core interaction with the consumer is key to create a long standing relationship with said consumer.With this line of reasoning, I was able to create interfaces which were minimalistic and modern.This couldn't have been achieved without the continued support of Mehdi, Elisey and Kareem who made sure that my designs were vetted and operational through days of testing and tinkering.

During work weeks, The team's energy was absolutely amazing in the sense that each of our strengths made up for each of our weaknesses which, in the end made us into one well oiled group. Through all the hours of coding and prototyping, we did not forget to have some fun along the way such as implementing Easter Eggs, watching a movie etc.

The magnitude of research I did for this project was more than I have done for any project. Thankfully, with learning sessions fashioned by Mehdi, my understanding of the subject was made crystal clear.From the structure of the EFOBE to the NP-Complete Problems, It was a learning journey which can only be experienced.

This research included learning about Bitcoin,Ethereum.LIteCoin,[Who could forget DogeCoin], Learning about the different consensus algorithms which is required to mitigate MITM or brute force attacks and also learning about NP Complete problems which included learning about the millennial problem, the differences between NP Hard and NP Complete and its individual and collective significance to science and mathematics.All in all, with the amount of research done,It was beneficial and useful to further this project to completion.

In the initial stages of this project, I had to improve my skills in Object Oriented Programming to create the required elements for this project to work.This involved brushing up my skills in classes, mathematical implementations in C sharp and GUI integration.This not only proved to be useful for the project, But it has also aided me in my regular courses at EPITA.

Tasked with the EFOBE structure planning, I made sure to do my research on the subject of Blockchain structures.So after two weeks of research, I was able to put forth a clear plan for the structure of EFOBE.After a few amendments made by the team at our weekly meet,The EFOBE was made and the project could begin with integrating the request handling and the NP complete problems which were tasked to the brilliant coders Mehdi and Elisey.

My main job was to create an intuitive user experience with simplicity in mind.I know I sound like Jony Ive right now,But the truth is that after working on this project, I found my niche in Front End Development. The website was created (amidst some delays) and features such as article templating and easy file uploading were added to bring the most up to date information to the people.

The website started off with nothing but an actual blank slate,But then I realised that the slate design is what i wanted to go for to achieve the simplicity required for the UI .So By adding a tile design throughout the website[36], The website was finally complete[37].But trust me when I say that once the end result was shown, the delay was worth it as it was probably one of the best websites that I ever made.

During the second phase, I was now tasked with the GUI and Web View portions of the project. Although the Web View was perfect and functioned as

[36] and of course..... Rounded corners!
[37] after being late by almost 10 hours

planned, I wasn't satisfied with the GUI which in its first iteration looked pretty bad. So for the final presentation, I managed to revamp the design to make it look in line with applications such as Slack and Google.

The intention of making the GUI was to allow the user to use EPICOIN without technical expertise hence facilitating for a broader audience.This comes in line with EPICOIN's goal of solving problems using Blockchain Technology. I sincerely believe that we have indeed achieved this goal.

The design theme was based off the designs of OnePlus Wallpapers and the animations of Google Search.With this ideology [38] I set out with the help of Elisey to create a new and borderless[39] application. After completing this, I finished off the integration of Kareem's Performance view and the CLI made by Elisey.In the end, we were left with an amazing interface comparable to Silicon Valley websites[Bragging is indeed one of my best suits].

The team has really been the best any guy could have asked for.Through all the bugs and problems we had to face, We did it together and I couldn't have asked for more. By creating a stellar Blockchain tasked with solving NP Complete problems, through the countless hours of coding, testing and of course GIT problems, ,we have proven the competency to be something more than a throwaway project just for grades.We have proven that we can create a product which is viable and scalable for the open market with a clear goal and solution in mind.

Running through the past presentations we had, I was aided by the team to create the most appropriate speech to dazzle the judges.Thanks to Kareem and his amazing persona, I was able to achieve this in no time.In fact due to his work, We were able to get the edge at the presentation by adopting a show first talk later approach which has worked for us time and time again.

Thinking about the final presentation,I can only be thankful to Elisey,Mehdi and Kareem for all the coding sessions, the meetings and of course the fun times we had. As we near the end of our project which we have completed,We hope to give the panel of judges a hell of a show and kick it out of the park.

[38] And kicking out some more bad design ideas
[39] ROUNDED CORNERS!!

8 Conclusion

8.1 What is the current state of Epicoin?

Epicoin is currently a proof-of-concept that shows that we can do block-chain differently and that there is more to this technology than just crypto-currencies or data storage.

At the same time, Epicoin is not a mature product yet. Its applications are still restrained and it will most probably not grow as a trend in the crypto currency world. Its main con is that it has a slow start since it is useless with a short EFOBE.

Nonetheless, Epicoin was fun and challenging, and we are satisfied with our work. We really enjoyed our journey as the Fundamentally Uncentralizable Cookies, and we are proud to say we have a strong team culture and spirit. We believe that we managed to make a strength out of our cultural and technical diversity, and that the goodwill and commitment of every single one of our members was a crucial factor in the completion of our work.

8.2 What is the future of Epicoin?

As we stated in our very first paper, we worked on Epicoin because we wanted to do something challenging with technology we know little about. Now that we spent the past fourth months on it, we will move to some others projects[40]. This does not necessarily mean that Epicoin will stop here, though. We released every piece of work we made under the MIT License since day one. If someone wants to take over our work and add their own touch to it, they are more than welcome to do so. Who knows? Maybe the Epicoin journey will continue through the efforts of others?

[40] For instance, some of our team members plan on participating in the IBM Call for Code challenge this summer

9 Bibliography

References

[20116] What is AES? A Brief History. Technical report, 2016.

[AEBCJC06] David Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. The Traveling Salesman Problem: A Computational Study. *The Traveling Salesman Problem: A Computational Study*, 1 2006.

[BM] Ingmar Baumgart and Sebastian Mies. S/Kademlia: A Practicable Approach Towards Secure Key-Based Routing. Technical report.

[cgi] cgi.csc.liv.ac.uk. Essay on PGP.

[Com] Compression Outline. Technical report.

[CSWH] IAN CLARKE, OSCAR SANDBERG, BRANDON WILEY, and W.THEODORE HONG. FREENET: A DISTRIBUTED ANONYMOUS INFORMATION STORAGE AND RETRIEVAL SYSTEM.

[EGL] R. Peter EGLI. TCP-Transmission Control Protocol TCP TRANSMISSION CONTROL PROTOCOL TCP-Transmission Control Protocol. Technical report.

[Exp] Expert Reference Series of White Papers. Technical report.

[HS] David A Huffman and Ire September. A Method for the Construction of Minimum-Redundancy Codes*. Technical report.

[Int] Inter-thread communication in Java. Technical report.

[Kat18] Katrina Kelly-Pitou. Stop Worrying About How Much Energy Bitcoin Uses, 2018.

[KC10] Pushkar Kolhe and Henrik Christensen. *Planning in Logistics: A survey*. 2010.

[KK] FORENSIC INSIGHT; DIGITAL FORENSICS COMMUNITY IN KOREA KEVIN KOO. Understanding of BitTorrent Protocol.

[LPGR] José Luis, Gómez Pardo, and Carlos Gómez-Rodríguez. The SHA-3 Family of Cryptographic Hash Functions and Extendable-Output Functions. Technical report.

[Mil09] Evgeny Milanov. The RSA Algorithm. Technical report, 2009.

[MMK74] Michael T. McClellan, Jack Minker, and Donald E. Knuth. The Art of Computer Programming, Vol. 3: Sorting and Searching. *Mathematics of Computation*, 1974.

[MPG12] Kelci Miclaus, Rob Pratt, and Matthew Galati. The Traveling Salesman Traverses the Genome: Using SAS® Optimization in JMP® Genomics to build Genetic Maps. 2012.

[Nak] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System.

[NJ] Java Notes and Achin Jain. JAVA NOTES-ACHIN JAIN-ASSISTANT PROFESSOR, CSE(NIEC).

[Pis05] V. I. Pistunovich. INTOR Design. *Soviet Atomic Energy*, 2005.

[POS80] J POSTEL. RFC 768 - User Datagram Protocol. Technical report, 1980.

[She18] Sherman Lee. Bitcoins' energy-consumption can power an entire country but eos is trying to fix that. *Forbes*, 2018.

[Tea] Teaching guide: Run-length encoding. Technical report.

[Tri] Gayatri Tribhuvan. A BRIEF INTRODUCTION AND ANALYSIS OF THE GNUTELLA PROTOCOL. Technical report.