EFG - Ecochain Financial Growth

Nikolaos Antonatos antonatos.nik@gmail.com ,
Akis Chalkidis akis@ecoc.io

November 11, 2020

Contents

1 Preface

The explosion of DeFi on Ethereum seems to bring a second chance for cryptomarkets. The bubble of 2017 discouraged many investors to get involved in cryptocurrencies in last years. Now we see the tides turning again. We believe that this is the right moment to expand the ecosystem of Ecochain.

EFG is the first ECRC20 token that will be used as a pioneer in DeFi on Ecochain. For this reason we decided to give it the name of EFG (Ecochain Financial Growth). It supports a decentralized application. Its purpose is lending, using ECOC as collateral. All logic is implemented in smart contracts. The system also contains oracles and open source clients for the user interface and interaction with the blockchain. Oracles take the prices in real time from sources that are already trusted by the public (coinmarketcap for example). The architecture is going to be explained in detail in section 6.

While initially the collateral will be only ECOC , the long term plan is to integrate other coins (and even tokens) from other platforms. The first platform will be Ethereum and its tokens, because of its userbase size. If demand for DeFi continues, then cross-chaining will expand to other big platforms.

2 Traditional Finance

2.1 Introduction

Financial sector belong to economics, which is the scientific study of ownership, use and exchange of scarce resources. For each field of economics there are different schools and theories. This is an unavoidable fact, because economics is a social science. In contrast to mathematics, social sciences can't be sure of the causes when studying an occurred event. For finance we are firstly going to present the traditional approach. Finance is interesting in the factors that individuals or companies consider to make a financial decision (buy, sell, hold or just predict a future event). Traditional finance school and theories makes a logical assumption: humans can think rationally. Also, humans want the best for themselves. Consequently, investors seek information, combine and process them using rational thinking and choose the best action or strategy that is the best of them.

2.2 Basic concept and terms

Traditional finance theorizes that best rational strategy for an investor is to choose the strategy that maximizes his *expected* returns and also minimizes his *risk*. Risk is a term in finance that is quantified and precisely defined. Risk is the variation around the expected value. Let's say that an investor has to

choose between two strategies σ_1 and σ_2 with expected returns of E_1, E_2 and risks V_1, V_2 perspectively. Then for:

$$E_1 = E_2, coice : min(V_1, V_2)$$

and for:

$$V_1 = V_2, coice : max(E_1, E_2)$$

The basic reason that rational investors prefer lower risk is that being closer to expected value can increase their ability to survive in the long run. Large financial institutes (for example banks) are willing to pay for lower risk, in other words they may prefer the strategy that has the lower risk even is the expected profits are smaller.

Now let us examine some other hypothesis of traditional finance. One basic hypothesis is the *Efficient Market Hypothesis* (EMH), which stated that the current prices of an asset reflect all public information. Assuming that there is no asymmetric information, then the assets trade at their *fair* values. Consequently no one can make a profit without taking a risk. In other words, arbitrage is impossible. Sellers can't sell higher of the actual (fair) value and buyers can't buy less than that at any time.

One other basic assumption that is made is that better have a value now than in the future. This is because there is a *risk free* return. The *capital asset pricing model* (CAPM) converts a future value to its today's value , which is always smaller. There is also the *Arbitrage Pricing Theory Model* (APTM), which is a multi factor math model based on linear equations. It takes as inputs macroeconomic indicators that reflect the systematic risk. In the microeconomic scale, the *Option Pricing Theory* (OPT) claims that not only spot prices of an asset but also fair values of derivatives can be calculated. Probably the most accepted and used in practice is the *Modern Portfolio Theory* (MPT), which tries to maximize the expected return of a portfolio for a specific risk level. We are going to present the above theories and hypothesis in more detail, starting from the most used ones.

2.3 The Modern Portfolio Theory (MPT))

We have already seen that rational investors are after high returns and low risk. The Modern Portfolio Theory - MPT - offers a tool to minimize risk on a expected profit level or vise versa, that is, to maximize returns on a specific risk level. The logic is simple; a portfolio contains many assets. Composing it with assets of different weights, which can be calculated using math, the minimum risk can be achieved. Except of the average (expected or E) ROI (return on investment) and variance V or standard deviation σ (risk measure), the correlation ρ between the assets are considered. The choice here to achieve the best result is to pick weights w_i that minimize the variance(risk).

The above are very common tools in finance. They also very well known for statisticians. We are going to present the math formula of them for a portfolio , just for completeness.
Expected return on investment:

$$E(R_p) = \Sigma w_i E(R_i)$$

Standard deviation (risk):

$$\sigma_p = \sqrt{\sum_i w_i^2 \sigma_i^2 + \sum_i \sum_{j \neq i} w_i w_j \sigma_i \sigma_j \rho_{ij}}$$

Correlation coefficient:

$$\rho_{i,j} = \frac{E[(i - \bar{i})(j - \bar{j})]}{\sigma_i \sigma_j}$$

While the Modern Portfolio Theory is not very modern today (it introduced in 1952) it is still widely used. In the next section we are going to see criticism on it but MPT and CAPM are used even by people who do not belong in traditional finance school.

2.4 The Capital asset pricing model (CAPM)

While MPT is used to compose or valuate portfolio investments, CAPM is a tool, which considers expected return and risk of an individual asset. CAPM takes into consideration the *risk free* rate , the *systematic risk* and the *unsystematic risk*. In traditional finance they believe that there is an asset who has a positive ROI and zero risk. They usually identify this asset as a short term positive state bond. So we have:

- *risk free* rate R_{rf}
- *systematic risk* R_m
- *risk of the asset* R_a
- *Coefficient* β_a between the asset and the market performance

and the formula:

$$R_a = R_{rf} + \beta_a * (R_m - R_{rf})$$

β is given by the equation:

$$\beta_a = \frac{Cov(R_a, R_m)}{Var_{R_m}}$$

For the cryptocurrency market, which we are interesting in, as systemic risk is we can safely assume a sharp decline in the prices of the leader coins (BTC, ETH) relative to fiat money or to a stablecoin. Because covariance ρ of all cryptoassets (except stablecoins) is close to 1, if the leader coins crash we expect with very high probability all cryptos to follow. Systematic risk in our case represent the probability of the whole cryptomarket system to fail.

2.5 The Efficient Market Hypothesis (EMH)

Efficient Market Hypothesis has a centric role and is a fundamental for traditional finance school. EMH supports the belief that for all assets the market price is very close to the actual(fair) price, at least in the long run. Consequently, no one can have a ROI above the free risk rate R_{rf} without taking a risk. In other words zero risk strategies can have a maximum ROI of R_{rf}. The above can be translated to equally other conclusions, for example active traders in the long run and on average will have a total ROI of R_{rf}.

EMH assumes that the above hold as long as there are no assymetric information. If some people have inside information then for them it is possible to achieve greater ROI than R_{rf} without taking a risk.

2.6 Arbitrage Pricing Theory (APT)

Many financial firms found that the use of CAPM (individual asset evaluation) for the MPT (portfolio evaluation) didn't actually work well in practice. In theory CAPM may be correct but fund manager soon discovered that using only historical data of the asset does not provide enough information. Many managers found that including macroeconomic indicators in the formula gave more precise results for asset evaluation. Today most managers in big firms have replaced CAPM for APT. Lets see the math form and explain the difference in more detail:

$$R_a = R_{rf} + \beta_n * (R_m - R_{rf})$$

The formula looks the same as the CAPM counterpart. The "only" difference is that β_a is replaced by β_n. Actually this is a huge difference which introduces much more complexity and difficulty. It is a "price" to pay to get more accurate results. β_a can be easily computed only by historical data of the asset price. On the other hand, β_n is computed by a linear equation containing not only β_a but also macroeconomic indicators with different weights. The difficulty of computing β_n come in two forms. Fist, the manager must decide which macroeconomic indicators to include in the linear equation and gather this data from trustworthy sources (for example, official statistics from the state). The second is to decide the weights w_i of the linear equation. Usually they use empirical methods (trial and error) or even artificial intelligence (AI).

Let's see an example. If the unemployment rate today in a country is much higher than before some years and the asset belongs to a sector that its products are of intensive labor nature then the asset is expected to underperform compared to its historical record. Or if the taxation changes the expected performance will change because ROI is the return after taxes. Or maybe the products of the asset belong to an industry that recently boomed, of the asset is expected to overperform etc. Including all these factors in the equation can give more precise predictions for the future value of the asset under examination.

For cryptomarkets it is logical to assume that it is heavily related to the real economy world. The difference is that the cryptoassets are not tied to a local economy but only to the global. For example events like pandemic, expectation of lower global economic output, uncertainty etc can have an impact on the prices as well.

2.7 Estimation of fair values - Option Pricing Theory (OPT)

Finally, it is time to speak about *derivatives*. Derivatives are financial products which are depend on the spot prices of the assets. Futures, options and swaps belong to this category. In traditional finance, their existence is fully justified. Because investors want to minimize their risk they can use derivatives for *hedging*. Hedging is possible when you take an opposite position equal to your current. For example if an investor in his portfolio has a high percentage of an asset then he is at risk (if the asset devaluates much). He can take a *short* position so whatever the outocme the portfolio will have a very low fluctuation.

Hedging can be used either for speculation or for hedging. According to traditional finance beliefs all investors are acting rational; so they all want to do hedging for their current positions and never speculate. This is still possible. Let us see an example.

Let's say that a bank A holds two currencies c_1 and c_2. It holds c_1 in much larger quantities because it needs it more (lets say its the local currency for the bank). Likewise, a bank B in a different country holds c_1 and c_2, with c_2 the bulk currency. Bank A is afraid that the value of c_1 may fall in the future. This exposes the bank A at great risk. Of course the bank could sell c_1 for c_2 but it needs large amount of c_1 to operate as it is the local currency. Likewise, bank B needs c_2 and is afraid that it will devaluate compared to c_1. Both banks can solve their problem with hedging; Bank A and Bank B agree to exchange c_1 : c_2 in a future date at a fixed exchange rate (*future contract*). That way they can bring exchange risk close to zero.

The above and all *derivative* products wouldn't be possible if the fair value couldn't be calculated. The difficulty of calculating the fair price is basically for options. They are computed as follows: According to the Efficient Market Hypothesis the (spot) prices of currencies or assets are trading to their fair values. Based on that, traditional finance assumes that the probability of the spot price to go up or down is equal (so it is 50% for each direction). And from that fact math formulas were formed. First was the *Black-Scholes* formula, which is the most used today. Other formulas are the *binomial option pricing* and *Monte-Carlo simulation method*. All these methods form the *Option Pricing Theory* (OPT), which states that it is possible to compute the fair price for all categories and types of options. We are not going to present more details on this subject because the purpose of this paper is not an analysis of derivatives.

We are only note that for cryptocurrency markets *Black-Scholes* formula can be used to calculate the fair price of any option.

2.8 Takeaway

From all the above the reader must keep in mind that in traditional finance the fundamentals are:

- All investors act rationally
- Rational behavior means following the strategy that maximizes the expected profit and minimizes the risk
- Free markets are efficient; the current prices are close to fair prices
- profits higher than the risk free asset can't be achieved with zero risk; expected higher profits carry higher risks
- Expected profits and risks can be calculated using formulas from statistics field. Investors can form a portfolio according to their desired risk level
- The fair value for all derivatives can be calculated; investors can use them as a tool to do hedging; that way they can change the risk level without the need to change their portfolio
- There is no speculation; investors are not willing to increase their exposure to risk (unwilling to have naked positions)

3 Behavioral Finance

3.1 Introduction

Some economists raised criticism on the fundamentals of traditional finance. They argued that the belief that all investors, and mostly individuals, make rational decisions, is simply not true. Individuals are human beings. To theorize that each of them gather all available data of assets, computes the expected profit and risk etc is something that doesn't happen in real world. Some of them may use financial consultancy but many others decide and act on their own.

From the above fundamental change of view the *behavioral economics school* was born. *Behavioral finance* belongs to this school. They mostly believe that while big institutions and professionals act or try to act rationally, the individuals usually don't. Even for big institutions(banks, mutual funds etc) there can be failure in management. Intentions or commands may not carry well by communication channels. Big corporations have many levels of management and commands from higher management may not be passed intact until they reach the low levels of hierarchy, where they will be implemented. But mostly

behavioral finance is focusing on individuals and why they may act irrationally, according to traditional model.

Another strong point against traditional school is their claim that public information is imperfect. Asymmetric information exists more often that traditional theory wants to believe. So even for investors who follow "rational investment" can't do so as they often luck sensitive information, because it isn't publicly available.

Finally, and most importantly, they don't adopt the Efficient Market Hypothesis. They argue that there are market anomalies for long periods of time, that is, current prices deviate too much from their fair values. They support their claims by research done on stockmarkets, which takes the historical data and the traditional formulas (CAPM, APT, OPT etc) and compares them for long time periods. They claim that EMH and empirical evidence do not match. This claim separates completely the behavioral thinking from the traditional one. Because of the importance of the matter, we are going to first see their objections on EMH and later analyze the reasons why individuals do not act rationally by the traditional school beliefs.

3.2 Objections on Efficient Market Hypothesis (EMH)

Most arguments against EMH are based on historical periods of stockmarkets, where the current value remained much higher or much lower from the fair price. Good examples are the "Great depression", started in 1929 and remained for too long. Prices where overevaluated and after the crash remained underevaluated for a long period of time. Other good example is the crash of 1987 in the US stockmarket. Maybe the most known is the "dot com bubble", where stock prices for tech companies were overevaluated for a long period of time and finally crashed sharply.

Other good point against EMH are *sessional patterns*. In most countries after new year celebration the values are increased without a reasonable explanation (for example in January in Unites States or February in China - Lunar new year).

Some research papers have shown that many investors are affected by the *Size Effect*. They prefer to invest in big companies and strong brands even if the ROI is significantly lower. They avoid medium size companies with high returns even when statistics calculate the same risk level. In other words they behave like consumers who prefer to buy products of well known brands even when the prices are higher than the competition.

There is an extensive bibliography and research of Efficient Market Hypothesis. We believe that what we presented here are already enough to show why

behavioral finance has objections on EHM. Next we are going to present why individuals do not actually act as traditional finance supposes to.

3.3 Why individuals may act irrationally

Behavioral economists believe that in real world individual investors are human beings, so their decisions are based not only on logic but also on their personal beliefs, culture, education, current mood and emotions etc. A good example for this is gambling. According to traditional finance there are no investors that will buy an asset if its expected value is negative (as they expect a loss). Following this logic, players who understand the rules of a game will never gamble, as all of these games have a negative expected return. This example shows that some investors are actually after high risk, because they want big profits even if they must risk big looses asymmetrically (more loses than gains in the long run).

Now we are going to present the most common reasons (according to behaviorists) that an investor's decision can be made against the traditional finance beliefs:

- *Overconfidence*
  Investors may have overconfidence. It may be in their character or because they had a recent big success (financial or of other nature). They believe that they can "see" or interpret the data better than the others.

- *Loss aversion*
  There are people who are willing to sacrifice the probability of high profits to avoid a much lower probability of losses. Consequently, they don't take the best financial decisions. These people have the exactly opposite mindset than of those who are inclined in gambling.

- *The disposition effect*
  The disposition effect is another form of loss aversion. According to this, investors sell more easily winning positions than losing positions. People who bought an asset higher than the current price do not sell, but wait until the price rise again to the old levels, so they can sell "without having a loss". Similarly they are hasty to sell an asset when the current price is above the price they bought because they will "make sure of a profit". They don't consider at all the current market data.

- *Inertia*
  Some people are very reluctant to take a decision, so they don't act or act very late. People who belong in this category usually luck self-confidence.

- *Framing*
  Financial data can be presented in different ways. Usually companies try to manipulate the decisions of investors. While they present the correct

data, they present it in such a way to look more attractive than they should (for example, they do comparisons of results with selected periods to make them look better than they are). Even objective announcements can be misinterpreted by investors, leading them to non rational decisions.

- *Failure in diversification*
  For some reasons many investors are failing to diversify their portfolio. They either compose it in such a way to have high risk or keep one or two assets only. There are many reason for failing. Most common is that people have a preference to an industry or category product. Consequently, they invest in these, having a false impression of low risk. But actually diversification is not possible if someone invests in one category only. It needs different assets with negative correlation to achieve a low risk portfolio.

- *Information misuse*
  This simply means that people do not understand what the numbers represent. Or they just miscalculate. Or, subconsciously, are influenced by some specific number. The reason for this are many. One reason is that people like round numbers more than floats. The second is that some investors are superstitious and have "lucky numbers". Other is cultural, where some numbers can represent good or bad luck (these numbers vary between countries a lot).

- *Conservatism bias*
  Conservatism bias is similar to loss aversion but is more general. There are individuals that are very conservative. Conservatism level depends on family beliefs, culture, sex and age among others. Conservative people tend to pursue zero risk levels. They will avoid very high expected profits even if acceptable risk level is involved.

- *Group behavior*
  People tend to mimic behavior of their peers. In the financial world "peers" can be other investors who hold the same or similar assets. It is also called the *Herd Instinct*. This phenomenon is visible in boom and panic periods. In boom periods, when individuals see a stock price (or industry index) rising, they are drawn into the market in a kind of *bandwagon effect*. So they buy without thinking what the fair value price of the asset is. Same thing is happening in panic periods. When prices are sharply falling many investors are rushing to sell at any price (even if it is close to zero). Again, they don't consider the fair price of the assets at all. This phenomenon appears often to cryptomarkets. It already happened many times. The most visible occasion happened in 2017.

- *Emotions*
  The mood is not same every day for an investor. A good or bad event changes temporary the view of a human being. A good event can bring optimism, while a bad one pessimism. Depending on the day, the investor

can see the prospects of an asset or portfolio different than it is. They are more willing to buy when they are in good mood and more willing to sell when they are sad or depressed.

All of the above are included in theories of behavioral economics. We are going to present in brief the most known of them.

3.4 The BSV model

Barberis, Vishny and Shleifer published their research in 1997. The subject was *market sentiment*. They claimed that they found strong indications against EMH. More specifically, they found that investors have an underreaction for assets that have a very short history record (less than a year) and overrreaction for long horizons (three to five years) of an asset. They claimed that the new information did not integrate properly in the current market price. Consequently, sophisticated investors could earn profits above the risk free rate by taking advantage of underreaction and overreaction, without bearing the corresponding risk. This claim is a blow in the face for EMH.

They tried to find the cause of the above. In their paper they argue that the causes are *conservatism* and *representativeness heuristic*. Conservatism, which already mentioned in the previous section, is the cause of underreaction. Conservative investors are afraid of the new assets and don't trust their financial date because of their very short history. Because of this they react "to little" or not at all. On the other hand, representativeness heuristic is the cause of overreaction. According to this, investors "see" patterns in financial records. These patterns do not actual exist (are meaningless) but the investors give them meaning. They believe that the data are trustworthy and carry much value and information because they are too long. Representativeness heuristic is a term in psychology and is supported by various scientists. Whether this is the actual cause of the overreaction or not, it very often exist for the new information in markets. Concluding, new financial information is not integrated properly or in time to the current values.

3.5 The DHS and DH models

In 1998, Daniel, Hirshleifer and Subrahmanyan published their paper. They tried to explain the above BSV findings of overreaction and underreaction for the markets. They accepted that overreaction and underreaction exist and argued that the reasons for this is *overconfidence* and *self-attribution bias*.

We already mentioned overconfidence above. Investors with overconfidence believe that they can extract more detailed information of the public data than the others. As a consequence, they overreact to news and financial information, causing price deviation from their fair value because they take action (buying or selling) in a non-rational decision process.

Self-attribution bias is in psychology the fact that a person tends to attribute successes to his own skills while his failures stem from external unpredictable factors (some refer to it as "bad luck"). DHS claim that they found empirical evidence that the higher the returns in a previous period, the more probable is that the investor agree with a statement claiming that his good performance (high profits) reflects his personal investment skills; and the opposite for low performance.

In 1999 , Hong and Stein adopt a different approach of BSV model. They emphasize the results of interaction between heterogenous agents. They define two different types of financial agents, the *newswatchers* and the *momentum traders*. They assume that both types are partly irrational because they process different sets of information. The newswatchers make forecasts based on signals that they privately observe about future fundamentals. Momentum traders consider past price changes. Their limitation is that their forecasts must be simplified functions that can use historical prices. Also, they assume that private information diffuses gradually across the newswatcher population. Underreaction comes from the gradual diffusion of information. Underreaction makes possible for momentum traders to enter the market and profit. Consequently, underreaction starts a circle of overrection.

There are other models as well for behavioral finance but the three above are the most known and accepted. Maybe the reader here believes that behavioral school was formed near the millennium, which is not correct.

3.6 The Prospect Theory

All the above were inspired from the *Prospect Theory*. In 1979, Kahneman and Tversky published their work "Prospect Theory: An Analysis of Decision Under Risk" which contains the first organized work on behavioral finance. The theory rewarded a nobel prize in 2002. Their theory contained loss aversion and *reference dependence*. Loss aversion, inertia and reference dependence try to explain the *endowment effect*. The endowment effect is the paradox that people are more likely to retain an object they own than acquire that same good or asset when they do not own it.

We already have explained loss aversion and inertia. But what is reference dependence ? When an investor evaluates an assets he has a subjective reference level. This reference level usually is the point that he bought or sold the asset in the past. Profit or loss then is compared to that level. Gains or loses are subjective because they depend of that specific personal point of the individual.

3.7 Historical examples that support behavioral economics theories

We already mentioned the great depression in 2009, the dot com bubble, the 1987 market failure. Actually, the stock markets are full of events of great failures (abnormal crashes) or unexplained boom periods. These boom or panic periods do not always reflect the real situation in economy. This supports behavioral economics theories for some economists. There are other stories, outside of stockmarkets, that seem to support these theories. The most know is the *Tulipmania*.

It is considered the first recorded speculative asset bubble in history. Here the assets where just flowers (tulips). Tulips, depending or their characteristics (shape, color etc) had different prices. The prices of rare (and not only) tulips skyrocketed in 1936. But let's explain in more detail what happened.

Tulipmania happened in Dutch in 1630s.Dutch was the richest country in the world at that time. The reason for this was sea trade and the colonies they had around the globe. Middle class emerged at the time. The first company was found. Commodity market (stockmarket for goods) was also found. Most Dutch people considered very rich according to the standard of other countries. The raise of middle class brought a tendency for luxury and expensive lifestyle. One of the luxuries was the ownership of very rare tulips (colors and shape). Blossom of tulips were very slow and difficult. Blossom needed 10 years while many plants failed to blossom. Also, to achieve rare color and weird shape the tulips should partially affected with a virus , which would change the genes of the tulip but would not be able to kill the plant. This was very difficult. So, not all tulips were the same. The expensive ones, which were in demand, were actually very scarce.

Finally a specialist on tulips managed to achieve a very rare tulip. From this tulip he had the bulb which could blossom after 10 years. He knew that after selling, other experts will plant and in the next generation this special tulip would stop to be rare. He wanted to keep the monopoly, so he sold the tulips only by the condition of the buyer to not reselling the plant or flower (be only a consumer). The selling were taken place in front of a notary. This happened before 1930 (in 1920s). As expected, some buyers didn't comply. After ten years (the next tulip generation) the monopoly of the rare species was converted to an oligopoly.

Until this time tulips were something like a jewelry. Actually the farmers didn't usually make a considerable profit. They had to buy a bulb at an expensive price, wait for ten years, and also risk for a non blossom event. At the same time , many traders, which belonged to clubs or guilds, where in haste to buy tulips or bulbs. As farmers wanted to normalize the risk they had and demand

was much higher than the supply, farmers though to sell with a discount to traders the tulip that will blossom in the future. This was done again with a contract, in front of a notary. So the price were decided today for the future product.

This, of course, is a *derivative*, a *future*. This is the first record in history that a *derivative* product was created. These contracts soon became anonymous because the traders wanted to buy or sell them easily. They were buying or selling them to other traders. Before the year 1936 only traders and farmers were interested in tulips. The contracts were sold between them every night and the prices were slowly gone up until the traders had the impression that much and easy money can come from these magic contracts.

The word past to the public. Common people, who didn't actually know anything about tulips start joining the trading guilds just to buy and sell later for higher. Prices started raising in parabolic. In 1936 prices skyrocketed. Some rare tulip contracts sold for a value of a house. Price went up so fast that attracted a very large portion of the Dutch population. The price always went up, never down. Anyone wanted a sure profit, fast and easy.

On 6 February , 1937, in the Haarlem city, in the guild's hall, which the trading was happening each night, some traders released that the prices were too high. They preferred to sell, even for a small discount, to "make sure of the profit". With the current level prices they would get rich without further risking. This night many traders wanted to sell for a discount. This fact scared many buyers. The prices start falling, causing a snowball effect. Suddenly the supply was very high, because the contract owners panicked seeing the price to fall so much in a couple of hours. Demand fell near zero as there were no buyers to buy. Until the morning, the contract prices fell around 95%. Haarlem owners did what was logic: they bought a fast horse and run to the nearest city to sell at discount, before word spreads out. They first newcomers sell at heavy discounts. Panic striked to those cities too. In 4 days the word spread in the whole country. Rare tulips rendered worthless. As a sidenote, it took 4 days because there were no advanced communication systems at that time(mobile phones or internet). With today's technology (digital markets) the bubble could burst in less than an hour.

The above example is used much from behaviorists because it supports much points of their theory. Inefficiency of the markets (the price was higher than the fair value for decades) and irrational behavior are clearly presented in the story. Overconfidence, inertia, framing ,failure in diversification , information misuse and herd instinct (buying at boom period and selling at panic) are all present.

3.8 Behavioral Finance in cryptomarkets

In this paper we are interesting in cryptocurrency markets. This paper does not want to reject the traditional school. But our belief is that in cryptocuurency markets behavioral theories describe better the investor's profiles. We believe that there are more speculators and fair prices are difficult or impossible to calculate for digital assets, whether they are coins or tokens. The investors are mostly individuals and no big institutions exist yet in crypto world as it happens to traditional markets. With this in mind we have designed our strategy for DeFi; the upcoming dApps target mostly to today's cryptomarket investors (individuals).

3.9 Takeaway

Of course the reader doesn't need to absorb the information of all these theories. He must just remember that there are theories that oppose the traditional school. They argue that investors can act irrationally or at least their behavior is unpredictable; the markets are not always efficient. And that traditional school theories contradict with recent empirical evidence of cryptomarkets.

4 EFG

4.1 Explanation

Here we are presenting the financial logic of EFG. In the next section we are going to present the technical details.

This dApp solves the problem for someone that holds crypto and needs liquidity to invest in cryptomarkets without the need to sell his assets. If he wants to keep his *long position* on the assets because he believes that they will gain in value then he can just use them as a collateral to get a loan. The system accepts ECOC as a collateral but later, as we have already said in the **preface**, other coins on different chains will follow.

When the user locks his asset he can receive an amount in EFG tokens. EFG is an ECRC20 standard and it has fixed total supply (of $1,000,000$ tokens). He must pay interest to the system for the EFG he received. Whenever he wants he can pay back his loan. The investor must return exactly the amount of EFG he borrowed plus the interest for the time period elapsed. The interest rate is per year and the total interest depends on the block timestamps.

Let's see the parameters of the system. These are the fixed interest rates on each asset and fixed safe margins. Let:

- c_{ec} be the amount of collateral of ECOC

- r_{ec} be the interest rate of ECOC in one year
- s_{ec} be the safe margin of ECOC/EFG
- t_0 the timestamp of the block that the loan started
- t_1 the timestamp of the repay block
- x_0 the exchange rate between ECOC and EFG (Oracles inject the exchange rate here) at borrow time

After locking at t_0 the user can withdraw EFG equal to $c_{ec} * s_{ec} * x_0$
The user can repay at any time. The interest is :

$$I = c_{ec} * s_{ec} * x_0 * \frac{r_{ec} * (t_1 - t_0)}{365 * 24 * 60 * 60}$$

because timestamp is in seconds. So the user must repay the amount rp

$$rp_{ec} = c_{ec} * s_{ec} * x_0 + I$$

$$rp_{ec} = c_{ec} * s_{ec} * x_0 + c_{ec} * s_{ec} * x_0 * \frac{r_{ec} * (t_1 - t_0)}{365 * 24 * 60 * 60}$$

$$rp_{ec} = c_{ec} * s_{ec} * x_0 (1 + \frac{r_{ec} * (t_1 - t_0)}{365 * 24 * 60 * 60})$$

There is a liquidation point. The system has the right to liquidate if the collateral fall below the current debt. Current debt rp_{ec} must be higher than the total collateral cl_{ec} in EFG value. So the system can liquidate only in the case when

$$rp_{ec_c} > cl_{ec} \Longrightarrow c_{ec} * s_{ec} * x_0 (1 + \frac{r_{ec} * (t_c - t_0)}{365 * 24 * 60 * 60}) > c_{ec} * x_c$$

Because the collateral is always positive , that is , $c_{ec} > 0$, we finally have:

$$x_c < s_{ec} * x_0 (1 + \frac{r_{ec} * (t_c - t_0)}{365 * 24 * 60 * 60})$$

where t_c is the current block timestamp and x_c the current exchange rate. The above condition is necessary (and is implemented inside the smart contract) to hold for the system to liquidate the position.

4.2 Uses

A natural question arises, where EFG can be used. EFG , as an ECRC20 token can be transferred freely. Consequently, EFG can be sold in the market. Or can be just being hold, if the owner believes that its price will raise in the future. It can be used to repay the loan, freeing up the collateral. Finally, it can be *farmed*, yielding another ECRC20 token, namely GPT (Grace Period Token). In section **5** we are going to explain the role of GPT in the system.

4.2.1 Speculation

As we have seen in the previous chapter many investors are after high risk and high profits. EFG is a good opportunity for them. If the ratio of ECOC/EFG, or more generally, any collateral to EFG goes higher then they can have a profit if the difference of exchange rate on the borrowed amount is greater than the total interest. Because they can repay back the collateral any time their profit prf at a time t is:

$$prf_t = \delta_{x_t} * c_{ec} - I = \delta_{x_t} * c_{ec} - x_0 \frac{r_{ec} * (t_t - t_0)}{365 * 24 * 60 * 60}$$

Of course the profit can be negative, so they may have a loss at time t. The risk for the investor is the liquidation. The closer in the liquidation point, the greater the risk of a forced liquidation (let us call this *margin call* point). The closer the safe margin of ECOC/EFG s_{ec} to 0 the less the risk is for the investor, but also the less they can borrow. The closer the safe margin to 1 the more dangerous. For this reason the dApp starts with a margin that is safe for the investor. We set $s_{ec} = 0.25$ for this reason.

4.2.2 Hedging

Hedging is when an investor bets on a position that is the opposite of his current one. Whatever direction the exchange rates take the investor is safe because the fluctuation (portfolio value) is low. Hedging is a strategy that rational investors follow (low risk and low expected profits). For the time being, hedging can't be carried out with only one pair. But in the future, when more coins are introduced to the system, a combination of collaterals can implement a hedging strategy.

4.2.3 Passive Income (staking GPT token)

The investor also has the option to temporary lock his EFG tokens. Doing this he receives a reward in another token(GPT). This action brings him passive income because GPT has a value. There is demand for it because it can be used to increase the grace period of a margin call (it delays the liquidation).

4.3 Initial distribution

It has already been mentioned that the supply is fixed at $1,000,000$ tokens. Ten percent(10%) is going to be withhold by the issuer and the rest ninty percent(90%) will be moved and owned by the lending smart contract. Anyone can get and withdraw EFG if he uses a collateral.

5 GPT

5.1 Description

GPT (Grace Period Token) is just used for delaying the liquidation of the collateral. It can also be farmed using **EFG**. The farming rate is $y_{GPT} = a_{EFG} * 1286 * 10^{-16}$ per second , where y_{GPT} is the yield (reward) and a_{GPT} the locked amount of **EFG**.As long as EFG is locked, GPT is given as a reward. GPT is an ECRC-20 token. Its total supply is fixed and equals to $10,000$ tokens. It can't be created on burned. GPT token will be a separate token (following ECRC-20 standard) but can be accepted from the lending smart contract.

GPT can be transferred freely (because it is an ECRC-20). It can also be used to delay the liquidation of a collateral. The amount to be used is equal to 5% of the debt in EFG (initial loan plus interest). It must be used before the liquidation point. Each grace period has a legth of 7 hours.

5.2 Initial distribution

The supply is fixed at $10,000$ tokens. Similarly to EFG token, 90% of the GPT token is initially be deposited to the lending smart and ten percent(10%) is going to be withheld by the issuer.

6 EFG technical information

We are going to present how the decentralized Application actually works. Non tech savvy investors can skip this chapter.

6.1 General architecture and logic

In traditional (centralized) applications there is *frontend* and *backend*. Backend, among others, has a database or some other persistent storage system. The access to backend is permissioned, limited to the owner of the application. But in decentralized applications the backend is the blockchain itself. Read access is permissionless, while writing (changing the smart contract state) must comply to the smart contract rules.

The smart contracts for the tokens do not specify any roles. But the lending smart contract defines the following roles:

- *owner*
  He is the deployer of the smart contract. His basic role is administration. He can change some parameters of the system and has also privileges on oracle authorization.

- *Oracles*
  Oracles are the only entities who are authorized to inject real world data to the smart contract. Specifically, they inject the exchange rates for each second(timestamp) of each pair of EFG (asset / EFG).

- *Pool founders*
  Pool founders are users who decided to invest a significant amount on EFG and create a pool. After the creation of the pool the users can join and borrow EFG. The profit for the founders is the interest of the borrowing EFG. They can also liquidate user positions if the collateral falls short.

- *Users*
  Users have the following options: depositing assets as collateral, borrowing, paying back, withdrawing or farming EFG. They can also delay the liquidation by using the GPT token.

The dApp consists of the following parts:

- *Frontend*
  Frontend is the client to be used by the common users. It consists of a web wallet and a User panel UI, which displays information about the current assets and has the ability to interact with smart contract (lending and staking).

- *Smart contracts*
  They are the "heart" of the dApp, implementing all the logic and saving all the necessary data on blockchain.

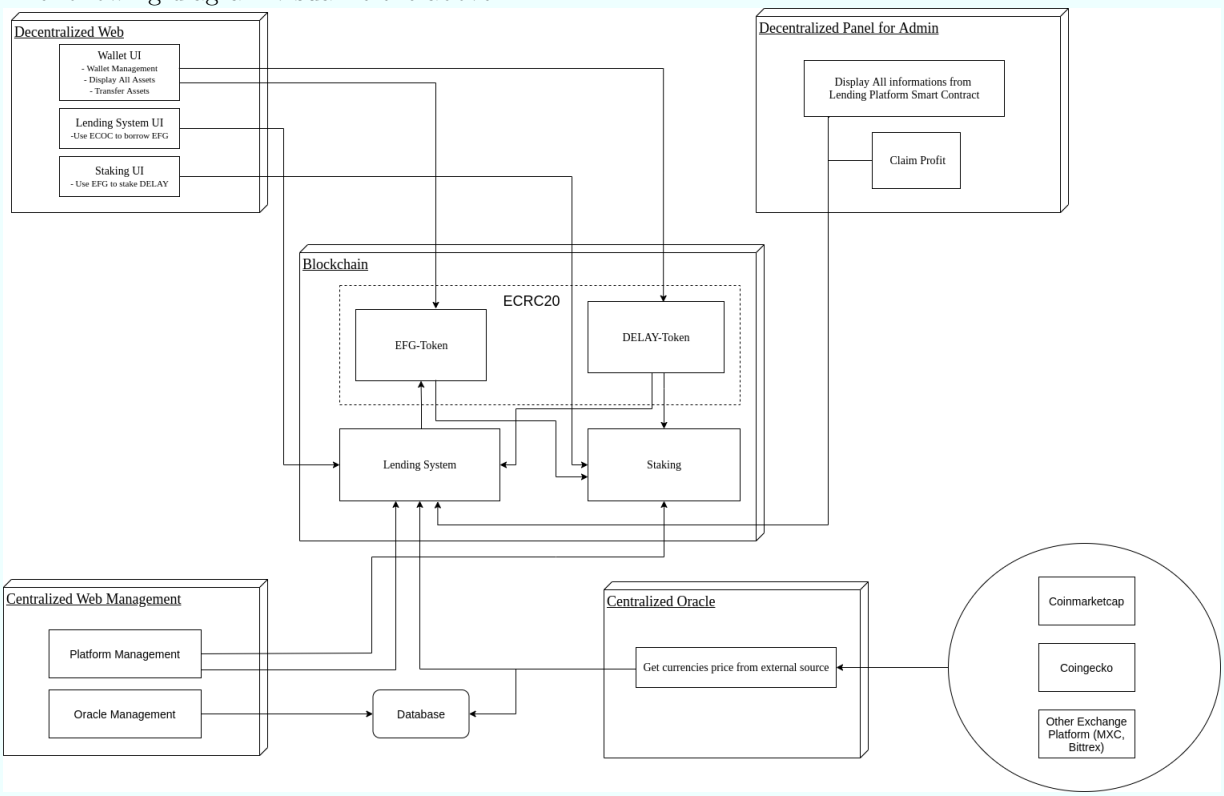
- *Oracles*
  Oracles are actually servers that gather information (exchange rates), store them in their database and inject them in the lending smart contract calling special function that only they have access. The oracles are *backend* only.

- *Admin panel*
  This is a frontend client, but only the admin can use it with his private key to interact with the smart contracts. He can check special statistics and call smart contract functions that only he has access (for example to replace an oracle).

- *Centralized management*
  This part is accessed by the dApp maintainers. It consists of frontend and backend. It watches the Oracles, if they are misbehaving it informs the admin to replace them. It can also gather and compute statistics for the dApp, feeding the admin panel. Finally, it can deposit tokens in case of emergency in the lending smart contract.

From all the above, only the first part is used by common users. The simple user is safe, because he interacts directly with blockchain and his frontend client isn't connected to any backend. His private keys are safe.

The following diagram visualize the above.



6.2 Smart contracts and their functions

We want to clarify some general things about smart contracts. Smart contracts are consisted by functions. Functions can be internal or external. Internal functions can't be called from outside,they are only internally used by smart contract. External functions can be called by anyone (if there are no restrictions) or by specific entities. We are going to present only external functions here, because these kind of functions are of the user's interest.

Another distinction we can make is between "read"and "write" functions. Read functions do not alter the state of the smart contract. For this reason they don't cost anything. In this occation we say that we are *calling* the function. Because ECOCHAIN is a permissionless blockchain, anyone can call these functions and get the returned data of the smart contract(without paying any fee). On the other hand , functions which alter the state of the smart contract are "writting" on the blockchain, so using such a function the user must pay a small fee (in ECOC). The use of such a function, if carried succesfully , is a valid *transaction*. In this occation, we are saying that we *send to* the function.

The user can use the official UI to interact with the dApp. But, let's not forget that EFG is trully decentralized. Consequentally, UI exist only for user-friendliness reasons. Someone, be it a programmer or a common user, can interact *directly* with smart contracts, skipping the UI entirely, if he so wishes. Programmers can use our SDKs, while common users can use their core wallet.

A word about Ecochain's core wallet. The user can see at left bar the smart contract tab. There are three option under this: create contract, send to and call. The user can use "send to" to carry a transaction or "call" to read the result from any call function. User must pass the correct arguments to each function. We strongly reccomend the use of official UI from users, especially those that are not tech savvy. We just mention the core wallet to prove how decentralized EFG is, dispell any doubt and give an extra option to advanced blockchain users.

Considering the above, it is clear that EFG is a decentralized application. Decentralization brings high security and freedom but also carries the burden of responsibility for the user. As there is no server, the private keys do not exist anywhere except the user's device. So, the user must be sure that he keeps his keys safe (proper backup strategy). There is no "username", "password" or "reset password" service. A loss of private keys inevitably results to permanent loss of the user's funds.

6.2.1 Tokens of EFG dApp

EFG and GPT tokens are ECRC20 compliant as they follow the ECRC20 interface. The following links show the smart contract source code for EFG and GPT:

- https://github.com/Defi-EFG/EFG-smart-contracts/blob/develop/lending/ECRC20/EFGToken.sol
- https://github.com/Defi-EFG/EFG-smart-contracts/blob/develop/lending/ECRC20/GPTToken.sol

It can be seen that the tokens are following the interface of ECRC20:

```
7  import "./IECRC20.sol";
8  import "./SafeMath.sol";
9
10
11 contract EFGToken is IECRC20 {
```

Because ECRC20 does not deviate from ERC20 standard (ethereum) and is very well known and documented, we are not going to spend more time analyzing it.

After launching the EFG application, any ECRC20 token can be added and accepted as a collateral. For this reason we can use wrapped tokens and atomic swaps. Tokens or native coins, which belong to other platforms (blockchains), can be imported in EFG dApp. The logic is that the user can swap the original asset with the wrapped token on ecochain, which follows the ECRC20 stanfdard, and join the application. After withdrawal the user can swap back his asset to the original chain. Using this strategy, EFG dapp can expand without any limitation.

6.2.2 Staking GPT smart contract

Staking smart contract is much simpler than lending. It consists of 2 read functions and 3 write functions. The purpose of this contract is to *farm* the GPT token by locking EFG. While the prevailed term in DeFi now is "farming" and "yielding" we preffer to use the term "staking" for two reasons. First, staking is more widely known while farming as a term is relatively new. The second reason is to stress the point of the tight connection (regarding functionality) that exists between EFG and GPT.

Before seeing the functions, let's have a quick look at the constructor of the smart contract. As a note, the source code can be found at StakingContract

Constructor:

```
20  function StakingContract (address _EFG_addr,address _GPT_addr) public {
21        mintingRate = 1286; /* mining rate per second in e-16 */
22        GPT = ECRC20(_GPT_addr); /* smart contract address of GPT , 4 decimal
               places */
23        EFG = ECRC20(_EFG_addr); /* smart contract address of EFG , 8 decimal
               places*/
24    }
```

The Staking smart contract is initialized by acceptinng the EFG and GPT tokens. Additionally, as it can be seen at line 21, the minting rate is set to $1286 * 10^{-16}$ per second. If the total EFG is locked then the GPT will be extracted in about three months. But in practise a significant portion of the EFG is going to be circulated in the markets. As a result the GPT minting period will be longer.

Send to functions:

```
42  function mintGPT(uint256 _amount) external returns(bool result)
```

mintGPT() accepts as input the EFG amount that user wishes to lock. In short, this is the function to deposit, lock and start minting GPT. The user must use approve() function first of the EFG token to approve the amount for the staking smart contract. We refer to this for explanation and education reasons only, because users which use the official UI do not need to do anything about approving (everything is done automatically under the scenes).

The transactiosn fails if:

- smart contract does not have any GPT left (all GPT is extracted)
- user does not have enough EFG (lower than the input argument) or the amount didn't get appoved prior from the user to smart contract

```
110 function withdrawEFG(address _beneficiar, uint256 _amount) external returns(bool
        result){
```

withdrawEFG() has the opposite effect of the previous function. Users can withdraw their EFG any time the full amount or partially and to any public address they wish(it can be different than the deposit address). For example, they can directly withdraw to an exchange. The user can withdraw any time, there is no restriction. Of course on withdrawal the GPT minting process is stopped for the user.
The transactiosn fails if the requested amount is higher than the deposited one.

```
75 function claimStakedGPT(address _beneficiar) external returns(bool result)
```

When claimStakedGPT() is called by the user the GPT that was minted is fully withdrawn to the desired address(again, the user can choose freely the destination address). This function can be called even after EFG has been withdrawn. In other words, the user can withdraw the GPT at any time he wishes.
The transactiosn fails if no GPT amount is left inside the smart contract (all GPT is already extracted by the users).

There are also two call functions:

```
143 function unclaimedGPT() public view returns(uint256 contractGPTBalance)
```

This function has only one purpose, to return the total balance of GPT for the smart contract. It is automatically called by the UI before the user sends to claimStakedGPT() or mintGPT(). The mentioned transactions will fail if the GPT balance is zero. By calling unclaimedGPT() first the UI saves gas for the user: if the functions returns zero then it just notifies the user and never initiate an unecessarry transaction.

```
134 function mintingInfo(address _beneficiar) external view returns(uint256
        lockedEFG, uint256 lastTimestamp, uint256 unclaimedAmount)
```

This function is used to extract info about the minting of a user. The locked EFG amount , time of last deposit and unclaimed GPT amount are returned. UI calls this function to get the data and display them to the user. It is costless.

The logic of this smart contract is clear. Its only purpose it for distributing the inital GPT to the users, proportionally to the amount they wish to lock

their main token (EFG). Next, we continue by explaining the functions of the lending contract.

6.2.3 Lending EFG smart contract

This smart contract is the heart of the dApp. Borrowing, repaying, liquidation and everything related to it exists here. The source code of the smart contract can be reached at github. There are 16 write functions and 17 read functions. Because of their number we are going to present the most in short, saving time for our analysis and explaination for the most important ones.

Roles Before continuing, we must present the roles that exist. There are four type of roles for the lending smart contract:

- contract owner
- Oracle
- pool founder
- common user

These roles are already mentiond in section §6.1

Contract owner privileged functions: We start by presenting the functions that can be triggered only by the contract owner. The role of the owner is to authorize/revoke oracles, set system parameters, authorize pool founding and allow token asset acceptance. Finally, consumed GPT belongs to the owner, so he can only withdraw it. Let's explain the privileged functions.

```
function addNewAsset(bytes8 _symbol, address _contract_addr, uint
    _collateral_rate)
```

Allows an asset to be accept by the dApp. The asset must follow the ECRC20 standard and be on ECOCHAIN.

```
function addNewPool(bytes8 _name, address _leader_addr, uint256 _EFG_amount)
```

Approves the foundation of a new pool (after the EFG was deposited by the founder).

```
function authOracles(address _oracleAddr, bool _action)
```

Activation/deactivations of oracles.

```
function setInterestRate(uint256 _interestRate)
```

Owner can edit the interest rate of EFG. Changing the rate has no impact for existing loans. The interest rate of a loan is constant and is set at the loan

creation time (first borrow). Only new loans are affected by a new rate. The user can borrow many times with the same old interest rate.

```
function setCollateralRate(bytes8 _symbol, uint256 _rate)
```

Collateral rate is the percentage of the asset value that is considered to compute the borrowing power (maximum loan amount that the user can borrow). It also play its role on liquidation condition. The collateral rate is set at first time that the asset is added. An exception is ECOC asset, which is set at the contract creation time , as it can be seen below:

```
function LendingContract (address _EFG_addr, address _GPT_addr) public {
    ..........
    /* Initial collateral rate of ECOC is 60% , 4 decimal places. */
    collateralRates["ECOC"] = 6000;
    }
```

We must clarify that, similar to setInterestRate() function logic, any changes affect *only* the new created loans. Old loans follow the collateral rates at their cration time until the end of their liftime (full repayment or liquidation). The reason is fairness for the users, because the rates are actually a crusial part of the initial greement between the contract and its users.

```
function withdrawGPT(address _beneficiar, uint256 _amount)
```

By this function the owner can withdraw the GPT that was used to trigger the grace period of loans. Any excess GPT that was deposited and never used by the users belongs to them. Users can withdraw the unused GPT by using another function, withdrawAsset().

Oracles privileged functions Oracles have only one privileged function because their only role is to inject the current rates between the assets to USDT, consuming third party APIs.

```
function setUSDTRate(bytes8 _symbol, uint256 _rate)
```

The function is simple, it sets the current exchange rate between the asset and USDT.

Pool founders privileged functions Pool founders are the entities that deposit capital(in EFG). They provide liquidity for the users and their profit derives from interest on EFG or at asset liquidation. They have the privilege to liquidate a loan under some conditions, which we are going to explain.

```
function marginCall(address _debtors_addr) external returns(bool result)
```

This is the liquidation function. It can be successfully triggered only by the pool founder where the loan belongs or by contract owner, as it can be seen in

the code:

```
require((msg.sender == l.poolAddr) || (msg.sender == owner));
```

Additionally, *all* of the following criteria must be met for the transaction to succeed:

- there is no grace period in effect:

  ```
  if(block.timestamp < l.lastGracePeriod) {
      return false;
    }
  ```

- the total collateral value has fallen short:

  ```
  if (totalDebt > computeCollateralValue(_debtors_addr) ) {
      return true;
    }
    return false;
  ```

  Collateral can fall short because of the accumilated interest or because the exchange rates where moved against the expectation of the debtor. Users must not borrow amount near to their borrow limit. Also, if they are near limit it is wise to fully repay the loan or use GPT to extend the grace period.

```
function increaseCapital (uint256 _EFG_amount) external returns(bool result)
```

This function can be used by the pool founder to increase the EFG capital inside the pool. If pool attracts many users then the EFG may not be enough for new members. Pool founder can increase the EFG in the pool if he so wishes. He must first approve EFG transfer from his account to the lending smart contract and then trigger the increaseCapital() function with the amount he wishs as an input. For the transaction to be successful, the approved EFG and balance of his account must not be less than the _EFG_amount which wants to deposit in the pool.

Rest of write functions All other function can be called by common users. Let's browse them one by one.

```
function depositAsset(bytes8 _symbol, uint256 _amount, address _pool_addr)
       external poolExists(_pool_addr) returns(bool result)
```

This function allows any user to deposit any asset(except ECOC). There are two requrements for the transaction to succeed. First, the loan should be in the initial state, meaning that must be unlocked(no borrowing has taken place yet). Second, the transfer of the asset must be carried out succesfully to the smart

contract.

```
function depositECOC(address _pool_addr) external
      payable poolExists(_pool_addr) returns(bool result)
```

This function is some as depositAsset, but is is only for depositing ECOC. The same reqirements hold. If the user tries to trigger this function without sending any ECOC the transactuon will fail.

```
function borrow(uint256 _amount) public returns(uint256 borrowedEFG)
```

Borrow function is the function to borrow for first time or continue borrowing as long as all required conditions are met. The conditions for the borrowing function to be successful are:

- The pool must exist:

  ```
      require(!(addressSearch(pool, poolAddr) == -1 ));
  ```

- borrowing power must exceed current total debt:

  ```
   require(_amount <= computeBorrowingPower(msg.sender));
  ```

- requested EFG amount for borrowing must not exceed the existing pool balance:

  ```
  Pool storage p = poolsData[poolAddr];
      require(_amount <= p.remainingEFG);
  ```

On first borrow the loan it initialized. At initialization the following actions take place:

- the loan is locked, make it unable to accept new collateral until repayd back (or been liqudated)

  ```
  if (firstBorrow) {
      l.locked = true;
  ```

- interest rate of EFG is saved. This protects the user after a change of the interest rates. As already mentioned, new rates can't affect loans agreed prior the change.

  ```
   l.interestRate = interestRateEFG;
  ```

- Other neccessary loan data are initialized.

On a succesful borrow() transaction the EFG will be automatically withdrawn out of the smart contract to the user. User can repeatadly borrow() as long as the above requirements are met.

```
function extendGracePeriod(uint256 _gpt_amount) external returns(bool result)
```

User can ask for a grace period or extend an already existing grace period. Each period is fixed in duration, which is 7 hours. While on grace period, the loan can't be liquidated under any condition. The cost for each grace period is 5% as can be seen in the smart contract:

```
uint256 constant private periodRate = 5; /* portion of debt in GPT to get the 7
    hours grace period , 2 decimal places (5%) */
```

The smart contract automatically counts existing GPT (it may exist as unused from a previous GPT deposit) and the deposited GPT. The amount must be no less than the minimum 5% of the total current debt, else will fail:

```
require(totalDebt * periodRate / 1e2 <= (l.remainingGPT + _gpt_amount) *1e4 *
    GPTRate / 1e6);
```

It is worth noting that if there is enough GPT from a previous deposit then this function can succeed even with an input of 0. That means that the user has the flexibility to call this function the first time depositing a big amount of GPT and later trigger the function without depositing again. After the loan ending (full repayment or liquidation) any unused GPT is available for the user to be withdrawn:

```
 balance[_debtors_addr]["GPT"] += l.remainingGPT;
```

```
function repay(uint256 _amount) external returns (bool result)
```

Similarly to borrow(), a user can repay the loan as often as he wants. In other words he can make a partial or full repay. This give the user much freedom. In fact the loan can be considered an "open loan", where borrowing and repaying can take place by any sequence, depending on the needs or perception of the user. This fact adds to user experience and satisfaction. If the repay is partial, then first the interest is decreased. If the interest becomes 0 then the capital is decreased. As a final notice, on full repay the loan gets reset, meaning that the interest and collateral rates are being reset. The user can change pool. Also, the new loan gets initialized again, so the new rates will apply. Of course after a full repayment the user has no obligation to borrow again. He can just withdraw all the released collateral and remaining GPT.

```
function withdrawAsset(bytes8 _symbol, uint256 _amount, address
    _beneficiars_addr) external returns(bool result)
```

This function is streightforward. It allows the common user to withdraw his collateral assets (except ECOC) after a full repay. The only requierment is the requested amount to not exceed his available balance. As an additional note, this function can also be used by pool founders or owner to withdraw any seized assets. In any occation, the caller of this function can decide to withdraw to any address. This gives him the flexibility to withdraw directly to an exchange for example, if so wishes. This function can also be used by pool founders to withdraw their profits from accumilated interest (EFG).

```
function withdrawECOC(uint256 _amount, address _beneficiars_addr)
      external payable returns(bool result)
```

Same as withdrawAsset(), but for ECOC only.

Callable (read) functions All of the following function are callable by anyone. They don't alter the state of the smart contract, so they can be called by anyone for free. While they can't change the dApp state they are very usefull, beucause the UI or the users or developers can directly access the data of the smart contract, display them to the user and also use them to correctly sent to the other functions (transactions). For example, canSeize() result can be used to decide if the marginCall function can be triggered, getPoolInfo() to decide if the user can trigger depositAsset() etc

```
function getAssetBalance(bytes8 _symbol, address _address)
      external
      view
      returns(uint256 assetBalance)
```

Returns the available amount for withdraw for asset with _symbol and for a beneficiar (precision is 8 decimal places).

```
function getPoolInfo(address _pool_addr)
      external
      view
      returns (
          bytes32 name,
          uint256 remainingEFG
      )
```

It accepts the pool founder's address as an input and returns the pool name and remaining EFG for the pool.

```
function getUserPool(address _depositors_addr) external view returns(address)
```

Returns the pool address for a specific user.

```
function getUSDTRate(bytes8 _symbol) external view returns(uint256 exchangeRate)
```

Returns the rate between an asset and USDT (precision is 6 decimal places).

```
function getAllAssets() external view returns(address[] allAcceptedAssets)
```

Return the contracts addresses of all ECRC20 tokens that are acceptable as a collateral for the dApp.

```
function listPoolUsers(address _pool_addr) external view poolExists(_pool_addr)
    returns(address[] members)
```

Returns an array of user addresses for a specific pool.

```
function getInterestRate() external view returns(uint256 EFGInterestRate)
```

Returns the interest rate of EFG (4 decimal places).

```
function getCollateralAmount(address _debtors_addr) external view
    returns(uint256[] collateralAmount)
```

Returns an array of amounts of the deposited collateral. It is used combined with getCollateralSymbols().

```
function getBorrowLimit(address _depositors_addr) external view returns(uint
    lendable)
```

Returns the maximum borrowing power (borrowing limit) in EFG. This amount depends only on deposited collateral and the current USDT rates. It does not influenced by the current debt. Precision is 8 decimal places.

```
function getEstimatedGPT(address _debtors_addr) external view returns(uint256
    GPTamount)
```

Returns the estimated GPT of a user. The return value can't be excact. The reason is that because of the nature of blockchain we don't know exactly when the next block will be formed, consequentally we don't know the exact block timestamp. The difference is negligible in practise. The UI of the dApp automatically adds a very small amount to the returned value to make sure that extendGraceperiod() will be called succesfully. Precision of returned value is 4 decimal places.

```
function getCollateralRate(bytes8 _symbol) public view returns(uint256
    collaterlRate)
```

Returns the collateral rate of an asset (4 decimal places).

```
function getDebt(address _debtor) public view returns(uint256 totalDebt,
    address poolAddress)
```

Returns current total debt of a user in EFG (8 decimal places). The returned amount has considered the borrowed amount plus the interest until the moment of the call.

```
function getAllPools() public view returns (address[] allPools)
```

Returns an array of addresses of the pool founders.

```
function getCollateralSymbols(address _debtors_addr) external view
    returns(bytes8[] collateralSymbol)
```

Returns an array of symbol assets of the deposited collateral. It is used combined with getCollateralAmount().

```
function getLoanInfo(address _debtor_addr)
      external view
      returns (
          uint256 amount,
          uint256 timestamp,
          uint256 interestRate,
          uint256 interest,
    uint256 EFGInitialRate,
    uint256 lastGracePeriod,
    uint256 remainingGPT,
          address poolAddr
      )
```

Returns useful information for a loan of the user.

```
function listLiquidable(address _pool_addr) external view poolExists(_pool_addr)
    returns(address[] allLiquidable)
```

Returns an array of all loans that can be liquidated for a specific pool. This function is used by pool founder to check which loans can be liquidated in his pool.

```
function canSeize(address _debtors_addr) public view returns (bool seizable)
```

The return value is a boolean. This means that the assets can be seized. This is a public function, meaning that it is used internaly by the contract and externaly by anyone he wants to check if a loan is liquidable at the calling time. It is used by the pool founders or owner before triggering marginCall(). Can be also used by the user. If the function returns true but the loan isn't liquidated yet then the user can try to prevent liquidation by using extendGracePeriod() function.

7 Conclusion

This paper will keep updating every time we add a chain or new features. For EFG dApp the goals in mind are:

- to increase the visibility of Ecochain to the world
- to bring liquidity on Ecochain
- to increase the userbase of Ecochain, boosting its utility(network effect)
- to grow the size of ecosystem, making easier for dApps to be created
- to give to the users the option to compose a portfolio

We hope that we were analytic enough to explain the motivation, goals,architecture and details about functionality for the dApp. If you have any questions don't hesitate to contact us. Email the EFG team at info@efg.finance

References

[1] [BSV] "A Model of Investor Sentiment", Barberis, Shleifer and Vishny, 1998

[2] [DHS] "Investor Psychology and Security Market Under and Over reactions",Daniel, Hirshleifer and Subrahmanyam, 1998

[3] [HS] "A Unified Theory of Underreaction, Momentum Trading, and Overreaction in Asset Markets",Hong and Stein , 1999

[4] [KT] "Prospect Theory: An Analysis of Decision Under Risk",Kahneman and Tversky , 1979

[5] [objections on EMH] "Irrational Exuberance", Robert Shiller's , 2000

[6] [MPT] Modern Portfolio Theory: https://www.investopedia.com/terms/m/modernportfoliotheory.asp

[7] [CAPM] Capital Asset Pricing Model: https://en.wikipedia.org/wiki/Capital_asset_pricing_model

[8] [APT] Arbitrage Pricing Theory: https://en.wikipedia.org/wiki/Arbitrage_pricing_theory

[9] [OPT] Option Pricing Theory: https://www.investopedia.com/terms/o/optionpricingtheory.asp

[10] Prospect theory: https://en.wikipedia.org/wiki/Prospect_theory