University of Twente.

Bachelor Assignment

Blockchain in Smart Grids

Author:

Tim van Genderen

Mentor:

V.M.J.J. Reijnders MSc

June 29, 2018

Abstract

Current cryptocurrencies like Bitcoin are a successful implementation of blockchain technology. However, blockchain can be applied to much more sectors, possibly the energy sector. In order to estimate whether this is possible a brief look at smart grids is taken and afterwards the working of a blockchain together with its security is analyzed. As of now, there are still some complications with applying a blockchain to a smart grid. There are some problems with the blockchain, like the mining energy, but also with energy, since energy is not something one can transfer from one place to another. On top of that, also the political complication of having a central entity exist. All in all, it might be possible to apply a blockchain in smart grid, but the complications are still there. Future improvement in blockchain technology and political adjustments will show if it becomes reality.

Contents

1 Introduction

The electricity network of, e.g. a city can be extended to a 'smart grid' by adding a ICT-layer on top of it. This layer of ICT makes it possible to receive real-time data about electricity usage and generation (e.g. solar panels) and also makes it able to steer certain assets, like batteries. Steering these assets makes it able to e.g. charge batteries at moments when the network is not that much used and thus not overload the network at moments when a lot of people are using energy. This steering can be done by letting the price of electricity depend on what time of the day it is (e.g. difference between day and night, paying more at peak moments). Considering that people are constantly using and generating electricity (and wanting to store this), there is the need to keep track of all the electricity transitions and one way to do this is by using blockchain technology. The main advantage of using blockchain technology is that no third party has to be involved when two people make an energy transition. Another big advantage is that data cannot be modified in any way possible. The blockchain technology is mostly known for its use in the Bitcoin, but also receives a lot of interest to use it in other ways, mainly as a tool for administration. There are a lot of startups experimenting with it in e.g. crypto-finances, ticket selling and other financial transactions.

Solving the problem of keeping track of electricity transitions could help the related problem of overloaded electricity cables. As mentioned before, at certain moments of the day there are peaks in the electricity usage which can overload the cables. This can cause a lot of problems since replacing them is expensive. This can be (partially) solved by levelling the electricity usage throughout the day, so effectively removing the peak(s), this method is called peak shaving. If keeping track of electricity usage/generation can be implemented using blockchain it is easy to see who is using electricity and if someone else has a surplus. In this case, it is possible for them to exchange electricity, instead of receiving electricity from the medium voltage power grid. This is better since the electricity has to 'travel' less through the cables, reducing the losses and therefore the amount of heat generated by the cables and transformer. Heat in those components can cause degradation which can damage the cables severely. In the worst case the cables would need to be replaced, which is very expensive.

The question is whether it is possible to use this blockchain method in a smart grid and what the possible difficulties are.

2 Smart Grids

A smart grid is essentially an electricity network with a layer of ICT on it. In order to fully understand what this means, a brief introduction to electricity and energy networks is provided.

2.1 Energy Transition

Energy transition is defined as the long-term changes in the energy landscape. The different changing trends in the energy landscape are discussed the following sections. The goal of energy transition is to make the life cycle of energy as clean and renewable as possible [58].

2.1.1 Renewable Energy

An important element to make the life cycle more renewable, is to use renewable energy instead of fossil fuels. Burning fossil fuels raises the carbon dioxide (CO_2) content in the atmosphere, leading to global warming. Global warming causes for example extreme weather conditions and rising sea levels to phenomena such as the ozone hole and less ice at the north- and south pole [40].

Using renewable energy instead of fossil fuels will reduce (or at least not further increase) the emission of CO_2 [41]. However, the production process of renewable energy does produce CO_2. For example, in the production solar panels CO_2 is emitted, but afterwards they will produce a lot of renewable energy. It is proven that this energy production cancels out the emission and a sustainable development can be created [37].

There are different kinds of renewable energy: photovoltaics, wind and hydroelectric power [58, 61].

1. **Photovoltaics (solar PV)**: Solar PV converts sunlight into an electric flow, which can be injected on an electricity grid. Unlike retrieving energy from wind or water, solar cells have a effectiveness. Current solar cells have an efficiency rate of 22,5%, so for future solar cells there is still room for improvement [2].

2. **Wind**: Wind power can be generated by letting the air flow through a wind turbine. Such a wind turbine is a tower with wings mounted on top of it.
   Wind energy and solar PV can be seen as complements of each other, since its production over the year is exactly the opposite. In the summer there are more sunshine hours with little wind, making solar PV more effective compared to wind energy. On the other hand, the winter has less sunshine hours and more wind, making wind energy more effective.

3. **Hydroelectric Power**: A third large producer of renewable energy is power generated by a water turbine, hydroelectric power. The most used type of hydroelectric power is building a dam to make an artificial lake. Its water will pass a turbine that generates electricity. Since this can be done in a controlled way by letting more or less water pass through the turbine, hydroelectric power is a consistent energy producer throughout the year.

2.1.2 Demand of Electricity

The demand in electricity increases, due to people relying more and more on electric devices. The current electricity demand and production will look something like Figure 1.

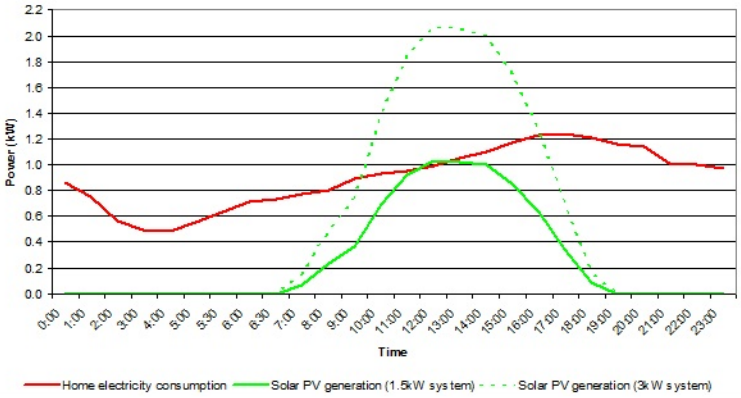


Figure 1: Electricity demand and solar production for a household in the summer [5].

This figure implies that at some periods of the day (in the afternoon) it is possible that a household produces more power than it uses, while on other periods of the day (e.g. in the night), it produces close to nothing and only consumes power. This uneven distribution is further strengthened if for example someone would charge their electric vehicle during the evening or night so they can use it the following morning.

Two examples that causes a large demand of electricity are heat pumps and electric vehicles [50, 58].

- **Heat Pumps**: Heat pumps are heating devices that force heat to move from one point to the other, sometimes the operation is reversible, meaning that it can also be used for cooling. Most known heat pumps are the boiler and air conditioning. Heat pumps are an alternative to fuel-fired heating and are more efficient that gas heating, how efficient they really are depends on the efficiency of the power plant. A power plant is an

industrial facility that generates electricity.
Even though this all sounds very positive, there are still some problems with heat pumps. In the winter for example, everyone wants to have a nice temperature in their house, meaning that the heat pumps will be used a lot. This demand coincides with the already existing peaks in the morning and evening, making this peak even higher. An opposite example will occur in the summer, in that case it is too hot and people are using a lot of air conditioning to cool their houses, also increasing the already existing demand peaks. One can imagine that if a lot of those heat pumps are clustered (e.g. in a city), that this will have a huge impact on the underlying electricity grid [59].

- **Electric Vehicles**: Electric vehicles (EV's) are a cleaner alternative to fuel based vehicles, since using electricity does not produce CO_2. Even though producing this electricity does produce CO_2 it is still less than the total emission of fossil fuels [51]. Some main problems of current EV's that hold backs people from buying them are the range, price and charging time. Current EV's, like the Tesla Model S 100D, can reach a range close to 500km, but are also very expensive [18]. On top of that, it takes at least a couple of hours to fully charge the battery of an EV [11]. If these conditions will become better and EV's are more accessible, charging all these EV's will become quite the challenge for the electricity grid.

2.1.3 Distributed Generation

Large power plants are concentrated at specific location (e.g. with a lot of cooling water and good access to the electricity grid). In contrast to that, the locations for renewable generators are very different, since these generators require other resources (e.g. non-overshadowed rooftop or a place with more wind), however there do exists places that combines multiple renewable energies (e.g. a combined solar PV and wind park). Another difference is that power plants are connected to the high voltage grid, whereas renewable generators are connected to either the low- or medium-voltage grid. This results in a decentralized way of generating energy (both topologically and in the grid), also called distributed generation (DG), resulting in some complications [17]. First of all, the electricity flows instead of only downwards (from power plant to the consumers) now also upwards (e.g. consumer with solar panels, injecting redundant electricity in the grid). Another problem is that it causes peaks in both directions, moments where people are consuming a lot and producing little (e.g. evening), but also moments where people are producing a lot and consuming very little (e.g. afternoon). Due to these moments where more electricity is produced than used, power plants are generating electricity while nobody can use it at that moment. This is why power plants would change their strategy and only produce at when needed, like at peak moments. This makes it less efficient (and a bit costly) for power plants due to constantly turning on and off.

2.1.4 Storage

Since the production of renewable energy does not follow the demand (it is mostly generated at moments where people do not use it), this energy needs to be stored somewhere or the behaviour of people should change. The storage can also compensate for varying supply and demand, thus the variable production of renewable energy can match the demand.
Current storage possibilities have some limitations. First of all, a storage has a certain capacity and can not store more than that. Also, a storage may have some limitations whenever it is used, for example maximal charge/discharge rates. On top of that, storing energy in the storage will result in a part of the energy being lost while charging or discharging [50, 58].

2.2 What is a Smart Grid?

All these trends causes the energy transition to be rapidly changing, and the electricity grid will face more and more challenges. In order to avoid this causing problems, it is useful to have some application on top of the electricity grid that can for example efficiently let electric devices charge at moments where the grid is not used much (e.g. in the afternoon when a lot of people are at work). A smart grid is this application and uses intelligent transmission and distribution networks in order to efficiently deliver the energy. The structure of an smart grid can be seen in Figure 2.

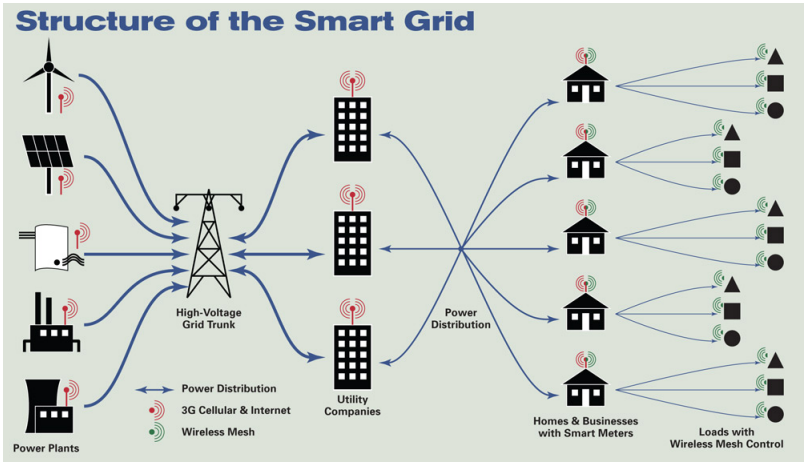


Figure 2: Example of the structure of a smart grid [28].

The main benefits of using a smart grid are the following [22, 39, 58]:

- **More balanced electricity demand profile**: As previously seen in Figure 1 of Section 2.1.2, the net electricity profile is not constant during the day and energy transition will make this more uneven. Due to the energy transportation loss following a quadratic relation to the current, a perfectly balanced demand profile throughout the day would cause the least amount of losses. A smart grid can help to make this profile indeed more balanced.
- **Islanding**: A smart grid can improve the flexibility of the electricity supply and enables the possibility to operate on an independent, disconnected part of the grid. This is called islanding and can be useful to do for some instances, mostly in places that are highly economically or life critical (e.g. hospitals), islanding can also be partial (e.g. having a small backup generator that can intervene at certain moments, but still using the network at peak moments). Energy transition creates possibilities for islanding such as using DG to provide energy instead of such a backup generator. A smart grid can for example optimize the time that DG needs to produce energy.
- **Local cooperation**: People who live close to each other can work together (share energy) in order for a more optimal use of the energy resources. An example of this is that it can be beneficial to charge the your EV at a lower rate when the neighbour's washing machine is running, this way the energy stream is being used more efficiently. Energy transition strengthens these kind of methods, since more and more electric devices want to make use if the electricity grid. A smart grid can enable the possibility to do this on a larger scale, also with the effect of balancing the demand profile.
- **Market Integration**: A practical smart grid needs to consider besides energy streams also value streams (e.g. money, but also comfort). This is due to the fact that in the energy market there are multiple stakeholders having different ambitions. These stakeholders are for example: the energy generation company, the energy supplier, the citizens who want to use the energy, and the grid operator. Currently the grid operator is responsible for the infrastructure of the grid, but are not considered to be part of the market. This may cause some complications if the smart grid would use a blockchain, since then the energy supplier would not be needed anymore. However, currently the energy supplier and thus indirectly the energy users has ties to the grid operator. More on this problem can be found in Section 5.

2.3 Current Smart Grids Concepts

A different view on a smart grid is that of smart coordination. Normally the fact that all energy resources are connected to the low-voltage network (this is also the part of the network where all houses are connected to) is seen as a problem since this part of the network has limited transport and high losses. However, since all this energy is so close to its users, it does not have to travel a long way from e.g. the power plant. Smart coordination is thus the view on the grid not as a distribution network, but as an infrastructure where people can share energy.

2.3.1 Demand Side Management

Demand Side Management (DSM) proceeds the idea of local cooperation, discussed in Section 2.2, by taking advantage of the flexible demand of the consumers [45, 58]. Currently people who want to e.g. charge their EV just plug in the power plug and the EV automatically starts charging. However, if this person uses the EV to travel to home, the EV will not be used until the next morning. So instead of immediately charging, it could be more beneficial to let the EV charge during the night, since during the time the grid is less loaded and will thus cause less losses. So if the customers declare their flexibility, at what time an electric device needs to be fully charged or when it needs to be finished (e.g. washing machine), then DSM can optimize the strategy for their flexibility. An example of an DSM implementation is Powermatcher [34].

2.3.2 Local Storage: Peak Shaving

In most countries there is a different tariff for the energy that is consumed and produced, people pay more use electricity then they will get for putting the same amount on the grid. Due to this difference, it is more beneficial to use self-consume the produced electricity. If at some point someone does not need the produced energy, it is a possibility to store it in a local storage. This local storage can charge at moments at which the grid is less loaded, and afterwards discharge at moments of load peaks. This will cause the peaks to be less high, hence the name of this process: peak-shaving [44]. An example of how the energy profile would look like, can be found below in Figure 3.

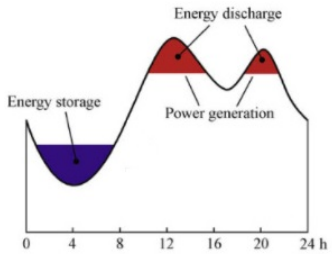


Figure 3: Energy profile; the effect of peak shaving [56].

3 Blockchain

Most people associate 'blockchain' directly with cryptocurrencies like Bitcoin. Even though this is the most known application, the technology of a blockchain can be used a many different ways. To give a simple idea of how the blockchain works: think of it as a subset of a database that keeps track of transactions (assets that are transferred from the owner to the receiver) in a very specific way.

The most important goal of a blockchain is to make 'the third person unnecessary'. For example, take the scenario where person A, having an account at bank X, want to transfer money to person B, having an account at bank Y, see Figure 4. In this case the bank X needs to check the balance of person A, make the transaction to bank Y and bank Y then transfers the money to the account of person B.
If this transaction was made in a blockchain, bank X and Y wouldn't be involved and the money can be instantly transferred from person A to person B.

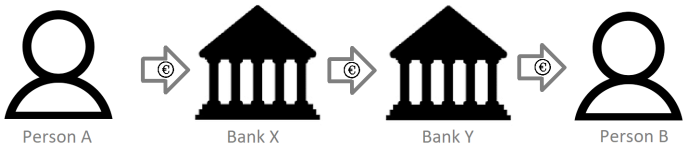


Figure 4: Current money transfer scheme.

Besides cryptocurrencies, blockchains are being experimented with in several startups in different fields: security, cloud storage, ticket purchasing and even contracts between doctor and patient [30]. The following sections provide some background information on how a blockchain works and its general advantages and disadvantages.

3.1 Structure of a Blockchain

As the name already suggests, transactions are grouped in blocks and these blocks are linked to each other where each block points to the previous block (except the first one, the so called genesis block). Each transaction requires a signature to verify the ownership, so each transaction is signed by the owner using the ECDSA [55, 66].

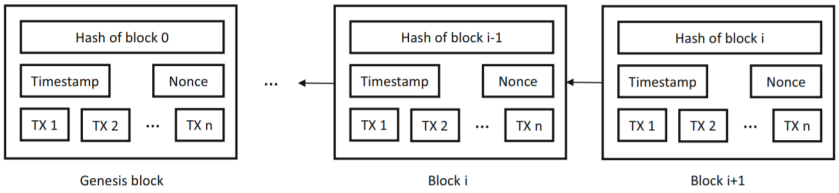


Figure 5: Schematic overview of a blockchain [43].

Each block contains a header that identifies this block. The elements of the header of a block are the following [43]:

1. **The hash value of the previous block**: This is the way a block points to the previous block, it hashes the header of the previous block.
2. **The merkle root hash**: One hash that represents all transactions, $TX1$ up to TXn, that are stored in this block. More details on how this hash is created can be found in 3.1.3.
3. **A timestamp**: The time at which the block was created.
4. **A target**: This target represents the difficulty of the block; a hexadecimal number of the same length as the created hash of this block, and the hash of the block must be less than this number in order to be valid. This number will be adjusted in such a way that the average mining speed will stay the same. More information about how this target represents the difficulty can be found in [62].
5. **A nonce**: A number that is used to verify the block, this nonce is adjusted until the hash value is valid. More details on the nonce can be found in Section 3.1.2.
6. Possibly a **version**. It is not necessary, but is for example contained in a block from Bitcoin [49].

The blockchain is stored on a lot of different places, called nodes, and it could be possible for everyone to download this data. After a block is created, it needs to be validated and added to the chain. There are different ways to do this, called consensus mechanisms and are further discussed in Section 3.2. In order for a block to be added to the chain, the majority of the nodes needs to agree on the validity of the block. A nice property of this structure is that even if a few nodes are compromised, they will not exceed 51% of the total computational power in the network, so their erroneous block will not be added to the chain [46, 57].

3.1.1 Implementation of a Blockchain

In order to better understand the working of a blockchain, an implementation of a slightly simplified version of it is made in Python. This section provides some pseudocode to show the working of it. The full implementation can be found in Appendix A.

Algorithm 1: Pseudocode for the working of a blockchain

Creating the first block;
genesis_block();

Ask the supplier of the transaction for details;
PopUp_Supplier();
ask for: name_supplier, name_receiver, amount;

Give the receiver a detailed message of the transaction;
PopUp_Receiver();
show: 'Name: name_receiver', 'Transaction: amount from name_supplier';
ask for: Yes **or** No;
if *Yes* **then**
    Show the supplier that the message has been confirmed;
    PopUp_Confirmed();
    Add transaction to the transaction pool;
else
    Show the supplier that the message has been denied;
    PopUp_Denied();
end

Put all transactions in the transaction pool in a new block and add it to the chain;
Function `add_block()`**:**
    Input : hash_previous_block, merkle_root_transactions, transaction_pool, timestamp, target, version
    Output: Add block to the chain
    Find correct nonce such that the hash value is lower that the target;
    return;

A print statement showing all contents of the blockchain;
show_all_blocks();

3.1.2 Nonce

A nonce is a random generated number used for verifying the block when the block would be added to the chain. A nonce gives 'originality' to a message (or block). An example that shows the usefulness of a nonce:
Consider the scenario where person A makes a purchase over the internet at a supplier. An attacker could intercept the encryption information of this ordering and (without needing to decrypt it) could send this over and over again, thus

ordering the product over and over again under the name of person A. With the usage of a nonce the encrypted message will be different for each ordering (since the nonce is different for each ordering), thus the supplier will discard the orderings with the same nonce.

3.1.3 Merkle Trees

One element of a block is the 'merkle root hash', this hash is obtained through a Merkle tree. The Merkle tree was proposed by Ralph Merkle in [42] and uses a tree structure to hash all the transactions into one hash [35].

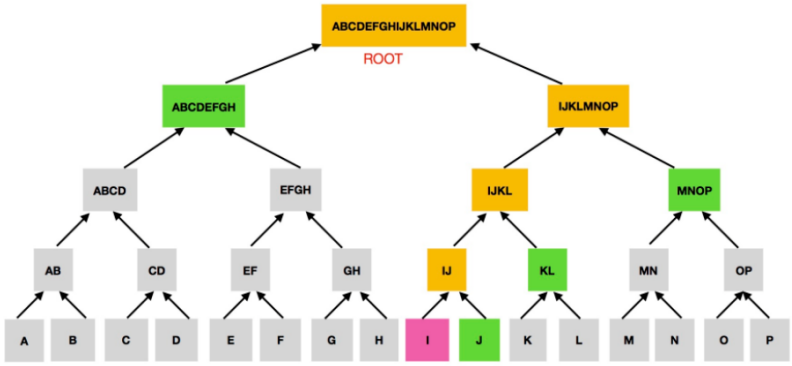


Figure 6: Diagram of a Merkle Tree and authentication path (highlighted in green) of I [14].

Let's say there is a list of 16 transactions: t_1 up to t_{16}, the first step is to hash all transactions individually into a list of hashes A up to P. Now the tree structure will be applied: the hashes A and B are concatenated into one long string, and this string will also be hashed into the hash AB. Afterwards, the same will be done for the pairs C and D up to O and P. Following the tree structure, AB and CD will be hashed together into $ABCD$ and so on. Finally this will result in the single hash $ABCDEFGHIJKLMNOP$, also called the root of the Merkle tree.

Of course the amount of transactions is not always a power of 2, so if in a branch of a tree there is an odd number of hashes, this hash is duplicated and then hashed. Take for example the example above where there are only 13 transactions, so hashes A up to M. The hashes $ABCDEFGH$ and $IJKL$ are unchanged, but hash M is alone, so it is duplicated and hashed into MM. In the next level MM is alone and thus duplicated and hashed into $MMMM$. Afterwards the root $ABCDEFGHIJKLMMMM$ can be calculated in the normal way. The amount of levels of a tree, given that the hashed transactions are at level 0, can be calculated by $log_2(N)$ rounded up, where N is the number of transactions.

In terms of blockchains, the merkle root hash is determined in the same way. However, this is not the only thing that is stored in the blockchain, also all calculated hashes in between (e.g. hashed transactions, CD, $IJKL$) are stored. This is done in order to verify the (hashed) transactions. Let's look at the same example of the 16 transactions, and transaction I needs to be verified, see Figure 6. In order to verify I, using the tree structure, the only hashes that need to be known are J, KL, $MNOP$ and $ABCDEFGH$, these are also called the authentication path. Now the root can be constructed and checked with the merkle root hash value inside the blockchain. This is why manipulating is very hard, in order to change one transaction, everything must be changed.

3.2 Consensus Mechanisms

A consensus mechanism is a set of rules to validate blocks and the state of the blockchain, that is agreed upon beforehand. Since all the nodes have the same consensus mechanism, the trusted third party is not needed anymore. Validating these blocks is called mining and is done by miners, more details can be found in Section 3.3. There are different consensus mechanisms where 'Proof of Work' and 'Proof of Stake' are the two most used ones [10, 16].

In **Proof of Work** (PoW) miners are competing to add the block, which can be done by solving an 'extremely difficult cryptographic puzzle': finding the nonce that makes the block hash valid. The first one in order to do so wins and receives a payment for his work (the block reward) and also receives a transaction fee. This transaction fee is a reward the sender of a transaction may add to the transaction to get priority for example. A remark on this is that even though there is no minimum transaction fee, at least a small fee is needed in order to get the block accepted [65].
Since a lot of different miners are competing to solve the puzzle and only the first one wins, computational power is a very important aspect. An advantage of this is that it is hard to cheat, since finding the nonce is already very time consuming. A big disadvantage of PoW is that it takes a lot energy due to the large amount of computations, making it environmentally harmful.

In contrast to PoW where miners are competing, in **Proof of Stake** (PoS) the creator of a new block is chosen in a deterministic way depending on their wealth. For example, in cryptocurrency this means the amount of currency they have, someone with 2 coins is twice as likely to be chosen than someone with only 1 coin. The chosen miner still has to add the block and receives a transaction fee for this, but no block reward.
The advantage of PoS is that it is uses way less energy, since only one miner is working to solve the puzzle. It also is very well protected against the 51%-attack, an attack where the attacker tries to accept erroneous block at the majority of the nodes (51%). This protection comes from the fact that it is very hard for a single node to obtain 51% of all existing currency. A disadvantage can be that since the most wealthy nodes are chosen and receive a reward if they are chosen, it can lead to a few nodes having almost everything and making it vulnerable

to e.g. DoS attacks. A DoS (Denial of Service) attack is an attack where a lot of different computers are trying to connect to e.g. a server in order to flood it with requests, making it unavailable for users.

There are several other consensus mechanisms, some interesting ones are Proof of Activity and Proof of Authority. **Proof of Activity** is a hybrid version of both PoW and PoS and makes the mining process easier, but validation harder. It tries to combine the best properties of both [4]. Another interesting mechanism is **Proof of Authority**, everyone can submit their transactions, but only a specific group (the authority) can verify them. This is very well suited for a private blockchain (see Section 3.4), it is not very energy intensive and very fast. The disadvantage is of course the sacrifice of trust to the authority.

3.3 Mining

Mining is the process of validating a new block, with the nodes trying to do this being called miners. Note that the amount of miners can differ from one (PoS), a few (Proof of Authority) to a lot (PoW). Miners can receive two sorts of reward for their work: in some cases a block reward for successfully being the first one to solve the puzzle (e.g. in PoW), and miners always receive a transaction fee (a reward the sender of a transaction adds to the transaction). Whenever a node finds a successful solution, this solution is verified by all other nodes and if at least 51% agrees added to the blockchain [15, 33, 64].

From a nodes' perspective mining starts with creating a candidate block. It lists a certain amount of transactions from the transaction pool, a list of all transactions that are not yet included in a block. This candidate block contains some information about the reward the node will get if it finds the right nonce, and a header with the elements explained in Section 3.1. One important thing that needs to be taken into account when creating a candidate block, is the size of the block. The size of the block mostly depends on the amount of transactions, since all other variables are more or less of a constant size. An increasing amount of transactions in a block will also increase the size of this block. Bitcoin has for example the threshold that the size can not exceed 1 MB [63].

After creating a candidate block, the real mining can begin: finding the correct nonce such that the hash of the block is less than the target. This can for example be done in the most intuitive way: incrementing the nonce by 1 until the hashed value is valid. An interesting thing to mention is that the starting nonce does not have to be 1, it can also be a random number. If a node has multiple computing devices, it is also to choose different starting nonces to make the process more effective.
In general there is no protocol for how to find a correct nonce, mostly due to the fact that the correct nonces are more or less random.

3.4 Different Blockchain Types

There are mainly three different types of blockchain: public, private and consortium (a hybrid of private and public) [9, 38].
A **public blockchain** is a blockchain where everyone can participate in every way possible. They can read it, send transactions and expect them to be verified, participate in the consensus mechanism. Its security relies, besides the cryptography, on economic incentives to create a large amount of nodes. Public blockchains are generally considered to be fully decentralized. Some advantages over a private blockchain:

1. Protection from the developers. There are certain aspects that can not be influenced by even the developers of the application. Two main reasons why this is actually beneficial for the developer: gaining trust and thus more participants, and they can not be pressured by some entity.
2. Network effects. Firstly, if multiple companies use the same blockchain, it will gain popularity. Secondly, it can also cut costs, Buterin gives the example of having a domain name system and a currency on the same blockchain, which can cut costs to zero by making a smart contract [9].

In a **private blockchain** writing and validating are restricted to a central organization. The read permission (what is stored in the blockchain) can either be public or (partially) restricted. Private blockchains are specifically useful for the internal parts of a single company, where the whole public does not have to know everything. Some advantages over a public blockchain are:

1. Changing the rules. Since a small group runs the blockchain, they can make adjustments (e.g. reverting a transaction) if they desire.
2. Cheaper transactions. New blocks only need to be validated by the small group instead of all participants.
3. More privacy. Since only certain people are allowed to read the blockchain, there is a greater level of privacy.

A **consortium blockchain** is a mix of the public and private blockchains: the consensus process is restricted to a specific set of nodes, the read permission is either public or restricted to the participants of the blockchain. A consortium blockchain is generally considered to be partially decentralized.

3.5 General Advantages

This section discusses the most important advantages of using a blockchain [23, 29].

The most important one is the main idea behind a blockchain: there is **no third party** involve, because of the decentralized nature of a blockchain. An example of this third party could be the government trying to interfere with the blockchain. For example, in the past the government has meddled with

several currencies, such as the German Mark, causing (hyper)inflation or other bad influences. No third party also leads to **financial efficiency** since people no longer have to pay fees or other costs to this third party (e.g. a bank). Even though they have to pay a certain transaction fee to get their transaction mined, this fee is still less than the credit card fee plus additional costs [6].

A blockchain also **reduces the risk of fraud/manipulation** because of the distribution and large amount of the nodes. This makes it hard for attackers to successfully manipulate data, because they have to use brute force attacks and do this for 51% of the nodes.

Another possible advantage is the fact that a blockchain is **immutable**, transactions can not be reverted. In some cases this is a slight disadvantage, but most of the time it is useful that transactions can not be reverted, e.g. for owner of the database since they can show that the data is not altered and thus reliable.

3.6 General Disadvantages

Besides these advantages, a blockchain also has some imperfections. This section covers the most important disadvantages of using a blockchain [23, 29, 52].

The most important disadvantage is that a blockchain is very **energy consuming**, mostly due to the consensus mechanism using a lot of computational power. For example, if the countries would be listed by energy consumption, Bitcoin (using PoW) is in 41st place and Ethereum (an other popular cryptocurrency, using PoS) would be in 72nd place. Also the amount of energy one Bitcoin transaction consumes is the same as for over 600.000 Visa transactions, one Ethereum transaction would equal over 45.500 Visa transactions [20, 21]. So even though the chosen consensus mechanism does make a large difference, even the more efficient ones still consume a lot of energy.

Another disadvantage is the **scalability**, the amount of transactions that can be stored in the blockchain per second. Currently Bitcoin has around 7 transactions per second (tps) and Ethereum around 15 tps, while Visa can handle over 24.000 tps [1, 53]. Depending on what the main goal of the blockchain is and thus how many tps it must handle, this can be a huge problem. Since this is an important bottleneck for cryptocurrency, developers are trying to solve this problem. Recently, the creator of Ethereum, Vitalik Buterin, proposed an idea called 'Sharding' that could solve this problem (at least for Ethereum). This could possibly result in Ethereum being possible to handle over one million tps [53].

Also a blockchain can have some **storing issues**. Since constantly transactions and thus blocks are added to the chain, the size of the total blockchain will only increase. At some point this can cause some problems, like being the blockchain too large that not all nodes can store a full copy of it, which can damage the security of the validation. Also, since not all nodes can store a full copy, only nodes with a lot of storage will remain, making the blockchain more centralized.

The following disadvantages will be mostly applicable to blockchain designed for (crypto)currencies. The anonymity of a blockchain may **attract criminals**. For example, there was a large digital black market running on Bitcoin. After a few years this black market was taken down [27], but it still shows the vulnerability to criminality. Another disadvantage of a blockchain for cryptocurrency is that they are very **volatile**, fluctuations of over 10% in one day are not that rare [12], thus making them very unreliable.

4 Cryptography behind Blockchain

Besides the structure of the blockchain being very well chosen, it must also rely on cryptography to be safe and secure. The blockchain uses two cryptographic concepts: a signature protocol and hash functions. This section provides background information on how these techniques work and why they are secure.

4.1 Hash Function

A hash function is a function that takes an input of arbitrary length and produces an output of fixed length. A simple example would be the modulo n operation, since it always produces an outcome of a number less than n. There is however a difference between this example and a secure hash function. A secure hash function h must have the following properties [47]:

- **One-way function**, also called preimage resistance. Given a hashed value $h(X)$ of a message X, it is computationally infeasible to compute X.
- **Second preimage resistance**: Given a hash function h and message X, it is computationally infeasible to find a second message Y such that $h(Y) = h(X)$.
- **Collision resistance**: It is computationally infeasible to find two messages X and Y such that $h(X) = h(Y)$.

Hash functions are considered to be secure until the contrary is proven. For example, SHA-1 and MD5 were two popular hash functions, but were taken down due to their vulnerability. At the time of this writing, the most trusted hash functions are the SHA-2 family [19]. More details about how these functions work and why they are secure can be found in [26].

4.2 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is a public-key cryptography that is based on the structure of elliptic curves over a finite field. The most known application is in Elliptic Curve Digital Signature Algorithm (ECDSA), a variant of the Digital Signature Algorithm (DSA) that uses an elliptic curve group in order to sign data in a safe way. ECDSA relies on a private key to sign a message and the public key, generated from the private key, to verify the signature. For example, Bitcoin uses ECDSA to sign the transactions that are made and afterwards stored in the blockchain. It is also used by several websites to sign their website [48]. First a brief introduction to modular arithmetic and groups is provided in order to combine it afterwards with elliptic curves.

4.2.1 Modular Arithmetic

An important aspect of a finite field is the modulo operation. This section goes over the most important operations in modular arithmetic. In general modular arithmetic over a number p converts a result into a number inside a specific range: 0 to $p-1$ [25].

Addition
Example: let $a = 10$, $b = 9$ and $p = 12$
Then $(a + b) \bmod p = (10 + 9) \bmod 12 = 19 \bmod 12 \equiv 7$.
Since $a + b = 19$ lies outside the range of $[0, p-1] = [0, 11]$, 19 is subtracted with $p = 12$ a certain amount of times (in this case only once) in order to get a result $7 \in [0, 11]$. $c \bmod p$ can also be explained as the remainder of the division $\frac{a}{p}$.

Multiplication
Example: let $a = 10$, $b = 7$ and $p = 23$
Then $a \cdot b \bmod p = 10 \cdot 7 \bmod 23 = 70 \bmod 23 \equiv 1$.
Since $a \cdot b = 70$ lies outside the range of $[0, p-1] = [0, 22]$, 70 is subtracted 3 times by 23 in order to get $1 \in [0, 22]$.

Multiplicative Inverse
The multiplicative inverse of b w.r.t. modulo p is the number $b^{-1} \in [0, p-1]$ such that $b \cdot b^{-1} \equiv 1$. b^{-1} only exists if b and p are co-prime, that is if $\gcd(b, p) = 1$. An example of this was actually given in the example of the multiplication operation, in that case $7^{-1} = 10$ and $10^{-1} = 7$ w.r.t. modulo p.

Division
The division $a/b \bmod p$ is defined as the multiplication $a \cdot b^{-1} \bmod p$, where b^{-1} denotes the multiplicative inverse of b.

4.2.2 Groups

A group is a set together with a function (a binary operation) that combines any two elements of this set into another element of that set in such a way that there exists an identity and every element has an inverse. The most known and used binary operations are (modulo) addition and multiplication.
The formal definition is as follows: Let G be a set together with a binary operation $\bullet : G \times G \to G$. G is a group under this operation, denoted as $(G, \bullet)$, if the following statements hold:

1. Identity. $\exists! e \in G$ such that $\forall g \in G : e \bullet g = g \bullet e = g$
2. Inverses. $\forall g \in G : \exists! h \in G$ such that $g \bullet h = h \bullet g = e$. This can also be denoted as $h = g^{-1}$ and $g = h^{-1}$.
3. Associativity. $\forall g, h, k \in G : (g \bullet h) \bullet k = g \bullet (h \bullet k)$

Some examples of groups are $(\mathbb{Z}, +)$, $(\mathbb{Z}_n, + \bmod n)$, $(\{1, 3, 7, 9\}, \times \bmod 10)$ and $(\{1, -1, i, -i\}, \times)$ (in the complex plane). Observe that $(\mathbb{Z}, \times)$ is not a group since fractions are not in $\mathbb{Z}$.

Each group has a certain amount of elements, this can vary from 1 (e.g. $(\{e\}, \times)$) up to infinity (e.g. $(\mathbb{Z}, +)$). The amount of elements of a group G is called its 'order' and is denoted by $|G|$.
Not only groups have an order, but also elements have an order. The order of an element $g \in G$, denoted by $|g|$, is the smallest $n \in \mathbb{Z}, n > 0$ such that $g^n = e$ (in additive notation: "$n \cdot g$" $= 0$). If this integer does not exist, then g has infinite order.

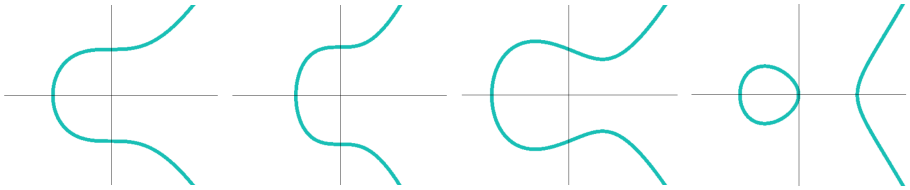
For example, take a look at the group $U(15) = \{k \in \mathbb{Z}_{15} \mid \gcd(k, 15) = 1\} = \{1, 2, 4, 7, 8, 11, 13, 14\}$ under multiplication modulo 15. Observe that the identity is the element 1. Since $U(15)$ has 8 elements, this group has order 8. In order to find the order of an element, for example 2, the sequence $2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 16 \bmod 15 = 1$ is computed, which results in $|2| = 4$. Not all elements have the same order, the element 4 has order 2, since $4^1 = 4, 4^2 = 16 \bmod 15 = 1$, and of course the element 1 has order 1 since it is the identity of the group [25].

4.2.3 Elliptic Curve

An elliptic curve is an equation of the form: $y^2 = x^3 + ax + b$. Two nice properties of an elliptic curve are:

- If a line intersects two points, it intersects a third point;
- If a line is tangent to the curve, it intersects another point.

Some examples of elliptic curves are:



(a) $y^2 = x^3 + 1$ (b) $y^2 = x^3 + 7$ (c) $y^2 = x^3 - x + 1$ (d) $y^2 = x^3 - x$

Figure 7: Some examples of elliptic curves.

In the context of ECDSA, a finite field can be thought of as a predefined range of positive numbers within every calculations must lie. This range is usually $0, 1, ..., p - 1$ where p is a (large) prime number, for this reason p is also called the prime module of the field. This field is $\mathbb{F}_p$, where every calculation is done modulus p such that the answer will lie in the range of $0, 1, ..., p - 1$. There is also a possibility to use $\mathbb{F}_{2^m}$, a binary field, as the finite field, more details on this can be found in [31, 32].

The definition of an elliptic curve E over $\mathbb{F}_p$ with $p > 3$ is an equation of the form

$$y^2 \equiv x^3 + ax + b \bmod p \tag{1}$$

where $a, b \in \mathbb{F}_p$ and $4a^3 + 27b^2 \not\equiv 0 \bmod p$, this condition avoids singularity. Then the set $E(\mathbb{F}_p)$ consists of all points (x, y) with $x, y \in \mathbb{F}_p$ that satisfy equation 1, and a special point $\mathcal{O}$: the point at infinity [31,32].

An example of how an elliptic curve would look like of it was plotted:

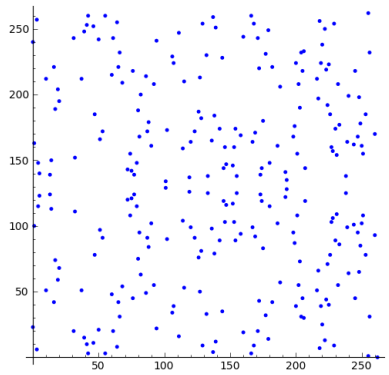


Figure 8: Plot of $y^2 = x^3 + 2x + 3$ over $\mathbb{Z}_{263}$ [3].

Now that a finite field in an elliptic curve is defined, the operations can be defined. Let's first sketch a visual idea of how these operations (addition and multiplication) are defined:

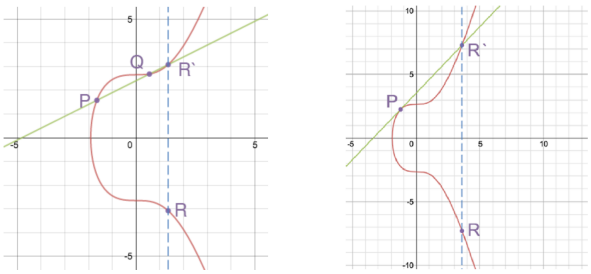


Figure 9: Point addition and doubling on an elliptic curve [48].

4.2.3.1 Point Addition

The first sketch of Figure 9 geometrically defines point addition: $P + Q = R$, where R is the reflection through the x-axis of R', with R' being the intersecting point of the elliptic curve and the line through P and Q.

The formal definition for point addition is as follows: Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ with $x_1 \neq x_2$ be two points in $E(\mathbb{F}_p)$. Then the sum of P and Q is denoted by $R = (x_3, y_3)$, where $x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2$ and $y_3 = \frac{y_2 - y_1}{x_2 - x_1}(x_1 - x_3) - y_1$ [31,32].

4.2.3.2 Scalar Multiplication

The second sketch of Figure 9 defines multiplication by 2, also called point doubling: $P + P = R$, where R is the reflection through the x-axis of R', with R' being the intersecting point of the elliptic curve and the line tangent to the point P. The formal definition for point doubling is as follows:
Let $P = (x_1, y_1)$ be a point in $E(\mathbb{F}_p)$ with $x_1 = 0$. Then $2P = (x_3, y_3)$ with $x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1$ and $y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1$ General multiplication, multiplication by a scalar a: $R = a \cdot P$, can be derived as multiple point doublings and point additions [31,32,48].
An example of multiplying point P by 7:

$$
\begin{aligned}
R &= 7P \\
R &= P + 6P \\
R &= P + 2(3P) \\
R &= P + 2(P + 2P)
\end{aligned}
$$

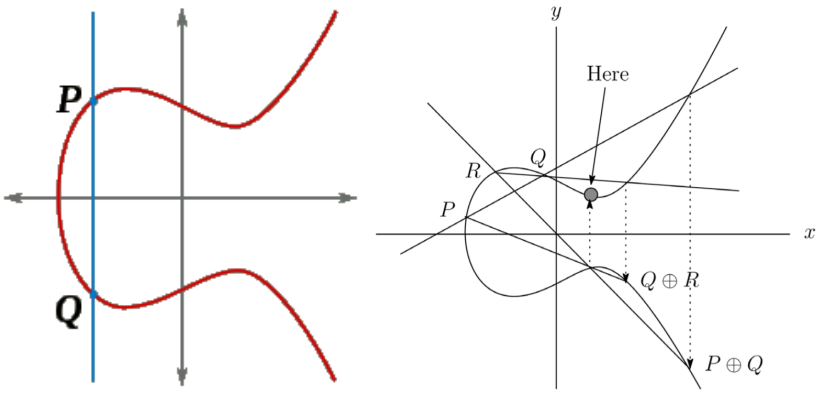
4.2.4 Elliptic Curve as a Group

From the previous sections one might raise the question: 'Is the elliptic curve in a finite field $\mathbb{F}_p$ together with addition (modulo p) a group?'. In order to answer this question, the three conditions of a group must hold: identity, inverses and associativity.

1. **Identity**: the point at infinity $\mathcal{O}$. Adding a point P to $\mathcal{O}$ (or the other way around) results in a vertical line through P and the reflection through the x-axis of P. However, point addition is defined as the reflection of the third intersecting point, which results in P.
   So $\forall P \in E(\mathbb{F}_p) : P + \mathcal{O} = \mathcal{O} + P = P$

2. **Inverses**: the reflection through the x-axis of the point, also called the complement point. Consider two points in $E(\mathbb{F}_p)$: $P = (x_1, y_1)$ and $Q = (x_1, -y_1)$, see Figure 10a. Q is the complement point of P, often denoted by $-P$. This results in a vertical line through P and Q, but also at infinity through $\mathcal{O}$.
   This gives: $\forall P \in E(\mathbb{F}_p) : \exists! Q \in E(\mathbb{F}_p)$, namely $-P$ such that $P + Q = Q + P = \mathcal{O}$
   Note that if $y_1 = 0$, P and Q are the same point. In that case P is doubled, which results in a tangent line, also intersecting $\mathcal{O}$. Here, P is its own inverse.

3. **Associativity**: Consider three points $P, Q, R \in E(\mathbb{F}_p)$, Figure 10b illustrates how the operations $(P+Q)+R$ and $P+(Q+R)$ result in the same point. A formal proof of associativity can be found in [24].



(a) Inverses (b) Associativity

Figure 10: Supporting sketches of elliptic curves for inverses [7] and associativity [24].

4.2.5 ECDSA

ECDSA describes an algorithm for signing data. There are some parameters that need to be fixed before ECDSA can be applied: the equation that is being used, the prime modulo of the field and a base point on the curve. This base point also has an order, see Section 4.2.2. The base point will be selected in such a way that its order is a large prime number. In fact, all practical applications of ECDSA will use large numbers for all these parameters.
There are several combination of parameters, the most known combination is 'secp256k1' for its use in Bitcoin. Secp256k1 uses the function $y^2 = x^3 + 7$ on the field $\mathbb{F}_p$ where $p \approx 1,16 \cdot 10^{77}$, a large prime number. More details on secp256k1 and other domain parameters can be found in [8].

Before messages can be signed, there first needs to be a private and a public key, which will be computed as follows:

- Generate a private key d: a random number between 1 and n, where n is the order of the base point G.
- Calculate the public key Q : $Q = d \cdot G$

Now, the algorithm for signing a message follows these steps:

1. Generate an integer k between 1 and $n - 1$
2. Calculate the point $(x, y) = k \cdot G$
3. Calculate r such that $r = x \bmod n$, if $r = 0$: go back to step 1 and find a different k
4. Hash the message that needs to be signed, this hash will be called z
5. Calculate s such that $s = (z + r \cdot d)/k \bmod n$, if $s = 0$: go back to step 1 and find a different k
6. The signature is the pair (r, s)

The algorithm for verifying the signature follows these steps:

1. Check that r and s are between 1 and $n - 1$
2. Calculate $w = s^{-1} \bmod n$
3. Calculate $u = z \cdot w \bmod n$
4. Calculate $v = r \cdot w \bmod n$
5. Calculate the point $(x, y) = u \cdot G + v \cdot Q$
6. Check whether $x \bmod n$ is equal to r. If this is true, then the signature is valid, otherwise the signature is invalid.

Note that everything is represented using hexadecimal strings. In the case of points on a plane, a point can be represented in two forms: compressed and uncompressed. The uncompressed form is one long string containing both the x- and y-coordinate of the point. The compressed form uses the advantage of the symmetry of an elliptic curve and only stores the x-coordinate. The compressed form also stores whether the point lies in the top or bottom half of the curve, which is indicated in the first byte by 02 or 03 (the uncompressed form always starts with 04) [8].

4.2.5.1 ECDSA: Symbol List

This list gives an overview of the most important used symbols in the ECDSA:

G	fixed	base point
n	fixed	order base point
d	private	private key
Q	public	public key
z	public	hashed message
(r, s)	public	signature

Table 1: List of symbols used in ECDSA

Where private means that only the signer knows this value, if it is public then everyone can see it, and fixed means that it is a parameter as discussed in the first paragraph of Section 4.2.5.

4.2.5.2 Correctness Verification Algorithm

It is not immediately clear why this verification algorithm works. The point that is calculated in step 5 (afterwards verified in step 6) needs to be equal to the point calculated in step 2 of the signing algorithm.

$$\begin{aligned}
(x, y) &= u \cdot G + v \cdot Q \\
&= u \cdot G + v \cdot d \cdot G \\
&= (u + v \cdot d) \cdot G \\
&= (z \cdot w + r \cdot w \cdot d) \cdot G \\
&= (z + r \cdot d) \cdot w \cdot G \\
&= (z + r \cdot d) \cdot s^{-1} \cdot G \\
&= (z + r \cdot d) \cdot (((z + r \cdot d) \cdot k^{-1})^{-1} \cdot G \\
&= (z + r \cdot d) \cdot (z + r \cdot d)^{-1} \cdot (k^{-1})^{-1} \cdot G \\
&= k \cdot G
\end{aligned}$$

This shows that the point calculated in both the signature and verification algorithm are the same and thus $r \equiv x \bmod n$ will too, making the verification algorithm valid.

4.2.6 Security

The ECDSA relies on the following security conditions [31, 60]:

1. The elliptic curve discrete logarithm problem (ECDLP) being hard. The ECDLP is defined as given an elliptic curve E over finite field $\mathbb{F}_p$, a point $P \in E(\mathbb{F}_p)$ of order n and a point $Q = k \cdot P$ where $0 \leq k \leq n-1$, determine the value of k.
   Observe that this calculation is used in step 2 of the signing algorithm. If ECDLP is not hard, then one can compute k and thus obtain the private key.
2. The generator for k (in the first step of the signing algorithm) is unpredictable.
   Otherwise one could predict k and obtain the private key from the signature (r, s).

3. The hash algorithm is a one-way function, which means that it is computationally impossible to retrieve the original message from its hashed value.
   If this was not the case, then one could obtain the original message and use its own key to sign the message in a valid way.

4. The hash algorithm is a collision-resistant function, that is, it is computationally impossible to find m_1 and m_2 such that $\text{HASH}(m_1) = \text{HASH}(m_2)$.
   Would this not be true, then one could ask for the signature for m_1 and falsely use the same signature for m_2.

Since there are no efficient ways to attack these problems, one would need to brute force these attacks in certain ways [60]:

- Shanks algorithm [36] breaks condition 1 within complexity $\Omega(\sqrt{n})$.
- Condition 2 would need 2^{160} tries before finding k.
- Random search [54] breaks condition 3 within 2^{160}.
- The birthday attack [13] breaks condition 4 within 2^{80}.

5 Applying Blockchain to Smart Grids

Now that it is clear what a smart grid is and how a blockchain works, the question is whether these two can be combined. In this case the blockchain will keep track of all the energy-transactions that are made. This section covers what the current problems are and, if possible, how these problems can be solved.

5.1 Blockchain: Mining Energy

As discussed in Section 3.3, the consensus mechanism in mining uses a lot of energy, even the more efficient ones where only one node mines the block. The difficulty of a block could be lowered in order to faster complete the mining process, but this would not reduce the time that the node is mining by much, since it just adds the blocks faster to the chain. Another slight disadvantage of this is that it would be a bit less secure, due to the right nonce being easier to find. As the difficulty decreases, also the level of security will slowly decrease, for this reason current cryptocurrencies increase the difficulty (e.g. Bitcoin does this about every 2016 blocks [62]).
This problem contradicts with the purpose of the smart grid to use energy as efficiently as possible. If the implementation would be possible in the future, one still needs to consider this energy problem and decide whether this loss in energy is worth it.

5.2 Blockchain: Amount of Transactions

Another possible complication could be the amount of transactions that need to be made. In order to perfectly balance the energy demand and production throughout the day, there would be a lot of transactions per second (in fact, it would be better to talk about milliseconds here). Current blockchains used for cryptocurrency can handle around 15 transactions per second, and if energy-transactions would be made at a higher rate, this would be a problem since the pool of transactions waiting to be stored would only increase. This said, recently a method is proposed that could make a blockchain store a million transactions per second [53]. If this would become reality, this complication would not exist.

5.3 Blockchain: The Miner

Whenever a block is mined, the miner will receive a reward. However, if the blockchain keeps track of energy transactions, the miner will get a reward in the form of energy instead of money.

Instead of basing the blockchain on energy, it could be possible to base it on money. This way the miners will get the reward in the form they want, the only change that needs to be made is that the transactions also need to be stored in terms of money. This can be a problem, since there most likely are different tariffs for using energy and putting energy on the grid, making it not actually a transaction since these values are not equal.

A possible solution to this is to have the blockchain based on energy, and have an extra layer on top of it that keeps track of how much each transaction is worth for both sides.

5.4 Future: Varying Energy Price

Proceeding on unequal transactions, in the future it is a possibility to let the energy price depend on e.g. what time of the day it is or the place where you live. This could be done in order to stimulate consumers to use energy at specific times (by making it time-dependent) or to make the transport cost included in the energy price (by making it place-dependent). This also has the goal of making it more attractive for consumers to charge/use their devices at non-peak moments. But in terms of blockchain the same problem will occur: it could be possible that the value of energy is different for both sides of the 'transactions'. The solution to this is the same as discussed in Section 5.3

5.5 Future: DSM

Currently people have to charge and use their electric devices all by hand, but in the future due to innovations like DSM, it could be possible to have a system that takes over the human acts. The question is whether DSM and a blockchain can work together. This would be no problem for the blockchain to handle. When someone declares their flexibility to DSM a future transaction is known. It is possible to make a template of this future transaction, but must not yet be added to the transaction pool. Only when the electric device is actually starting to charge or being used, then the transaction will be added to the transaction pool. DSM would have no influence on the applicability of the blockchain.

5.6 Energy: Can not be controlled

Energy is not something one can just transfer from one place to another. The energy flow is a result of changing voltages in the grid. When someone 'inserts' electricity into the grid, at that place the voltage level rises. Since the grid wants to have a constant voltage level, it will go to places with a lower voltage level, like for example someone who needs the energy. This means that an energy transaction does not actually transports energy from one place to the other. It is unsure whether this will be a problem, but if so, there is nothing that can be changed since physics can not be beaten.

5.7 Energy: Grid Operators

Currently every consumer has a contract with their energy supplier, and every energy supplier has a contract with the grid operator. The grid operator is responsible for the infrastructure of the grid (e.g. repairing cables). If a blockchain would be used in the smart grid, the 'third person', being the energy supplier, is not needed anymore. However, there still needs to be a connection to the grid operator in some way, but how?

A possible solution could be that instead of a contract with the energy supplier, consumers could have a contract with the grid operator. Another possibility could be to make the grid operator a consumer in the blockchain.

5.8 Law: Central Entity

By the Dutch law, it is necessary to have a central entity that checks everything like reading out the meters. If this central entity would not exist, every consumer can manipulate the meter, making the system unreliable. Currently this central entity is the energy supplier, but with the introduction of the blockchain, they are removed. In some way there needs to be a central entity, but on the other hand a central entity checking everything is exactly the opposite of the main goal of introducing the blockchain.

6 Conclusions

This paper gives an overview of smart grids and blockchains together with its security. An energy network is explained together with the energy transition and how smart grids are a solution to the current trends in the energy landscape. Afterwards the working of a blockchain is analyzed and the advantages and disadvantages of its usage are discussed. Also a short look into its security was taken by analyzing the cryptography behind it.

Applying a blockchain in a smart grid in order to keep track of all the transactions that are made, is currently not possible but maybe in the future it will. There are some complications with implementing a blockchain.
The main one is the energy it takes to add new blocks to the chain. Even with the most energy-friendly consensus mechanisms, the energy consumption is still significant. Also the amount of transactions can matter, since it maybe needs to handle hundreds of transaction per second. Current blockchains can not do that, but a current new method may solve this problem.
An interesting development is the possibility to let the energy price vary per consumer. However, this does not need to be a problem for the system, since an extra layer can be added that keeps track of the value in terms of money for both the supplier and receiver of the energy transaction. Another current development is demand side management and should also be no problem for the blockchain to handle. Whenever someone declares their flexibility, the transaction will not be added until the device actually starts working or charging.
When talking about energy transactions, this transaction is actually a result of changing voltage levels and thus can not be forced to go in a certain direction, making it not actually a transaction. Whether this will be a problem is unclear, but if so, there is nothing that can be changed about it.
Currently, the consumer has a contract with its energy supplier and thus indirectly with the grid operator. Using a blockchain would remove the energy supplier in this picture, but one must not forget about the grid operator. Possible solutions to keep in touch with the grid operator is to let the consumer make a contract with the operator, or to make the operator a consumer in the blockchain. Another political problem is the central entity. By the Dutch law it is necessary to have a central entity, which is now the energy supplier, that checks everything. Implementing a blockchain would remove this central entity, but if the central entity would be added in another way, it would actually make the goal of a blockchain irrelevant.

References

[1] N. Acheson. Can bitcoin scale? https://www.coindesk.com/information/can-bitcoin-scale/, 2018. Accessed June 26, 2018.

[2] V. Aggarwal. What are the most efficient solar panels on the market? https://news.energysage.com/what-are-the-most-efficient-solar-panels-on-the-market/, 2018. Accessed June 27, 2018.

[3] J. Bauer. Ecc tutorial. https://www.johannes-bauer.com/compsci/ecc/. Accessed June 20, 2018.

[4] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake. *ACM SIGMETRICS Performance Evaluation Review*, 42(3):34–37, 2014.

[5] K. Bhavnagri. Home energy consumption versus solar pv generation. https://www.solarchoice.net.au/blog/home-energy-consumption-versus-solar-pv-generation, 2010. Accessed June 27, 2018.

[6] D. Blystone. Bitcoin transactions vs. credit card transactions. https://www.investopedia.com/articles/forex/042215/bitcoin-transactions-vs-credit-card-transactions.asp, 2018. Accessed June 29, 2018.

[7] Brilliant.org. Group theory. https://brilliant.org/wiki/group-theory-introduction/. Accessed June 20, 2018.

[8] D.R.L. Brown. Standards for efficient cryptography 2 (sec 2). www.secg.org/sec2-v2.pdf, 2010. Accessed June 19, 2018.

[9] V. Buterin. Vitalik buterin: On public and private blockchains. https://www.coindesk.com/vitalik-buterin-on-public-and-private-blockchains/, 2015. Accessed June 25, 2018.

[10] A. Castor. A (short) guide to blockchain consensus protocols. https://www.coindesk.com/short-guide-blockchain-consensus-protocols/, 2017. Accessed June 25, 2018.

[11] Clippercreek. How long does it take to charge an electric car? https://www.clippercreek.com/charging-times-chart/, 2018. Accessed June 28, 2018.

[12] CoinMarketCap. Historical snapshot - march 18, 2018. https://coinmarketcap.com/historical/20180318/, 2018. Accessed June 26, 2018.

[13] D. Coppersmith. Another birthday attack. In H.C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 14–17, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[14] D. Cosset. Blockchain: what is in a block? `https://dev.to/damcosset/blockchain-what-is-in-a-block-48jo`, 2017. Accessed June 26, 2018.

[15] D. Cosset. Blockchain: What is mining? `https://dev.to/damcosset/blockchain-what-is-mining-2eod`, 2018. Accessed June 26, 2018.

[16] D. Cosset. Quick overview of consensus possibilities in a blockchain. `https://dev.to/damcosset/quick-overview-of-consensus-possibilities-in-a-blockchain--2d24`, 2018. Accessed June 25, 2018.

[17] E.J. Coster, J.M.A. Myrzik, B. Kruimer, and W.L. Kling. Integration issues of distributed generation in distribution grids. *Proceedings of the IEEE*, 99(1):28–39, Jan 2011.

[18] EV Database. Electric vehicles with longest range. `https://ev-database.uk/compare/electric-vehicle-longest-range`, 2018. Accessed June 27, 2018.

[19] R.L. Chavez Diaz. *Blockchain: a Technical Overview and Applications beyond Cryptocurrencies*. University of Twente., 2004.

[20] Digiconomist. Bitcoin energy consumption index. `https://digiconomist.net/bitcoin-energy-consumption`, 2018. Accessed June 26, 2018.

[21] Digiconomist. Ethereum energy consumption index (beta). `https://digiconomist.net/ethereum-energy-consumption`, 2018. Accessed June 26, 2018.

[22] H. Farhangi. The path of the smart grid. *IEEE Power and Energy Magazine*, 8(1):18–28, January 2010.

[23] W. Fauvel. Blockchain advantage and disadvantages. `https://medium.com/nudjed/blockchain-advantage-and-disadvantages-e76dfde3bbc0`, 2017. Accessed June 26, 2018.

[24] K. Fujii and H. Oike. An algebraic proof of the associative law of elliptic curves. *Advances in Pure Mathematics*. pages 649–659, 2017.

[25] J.A. Gallian. *Contemporary Abstract Algebra*. Cengage Learning, 2013.

[26] H. Gilbert and H. Handschuh. Security analysis of sha-256 and sisters. In M. Matsui and R.J. Zuccherato, editors, *Selected Areas in Cryptography*, pages 175–193, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[27] A. Greenberg. Prosecutors tract $13.4m in bitcoins from the silk road to ulbricht's laptop. `https://www.wired.com/2015/01/`

prosecutors-trace-13-4-million-bitcoins-silk-road-ulbrichts-laptop/, 2015. Accessed June 26, 2018.

[28] Homepower. Structure of the smart grid. https://www.homepower.com/structure-smart-grid, 2018. Accessed June 29, 2018.

[29] World Crypto Index. Advantages and disadvantages of decentralized blockchains. https://www.worldcryptoindex.com/advantages-disadvantages-decentralized-blockchains/. Accessed June 26, 2018.

[30] C. Jee. Blockchain startups to watch. https://www.techworld.com/picture-gallery/startups/blockchain-startups-watch-3669058/, 2018. Accessed June 24, 2018.

[31] D. Johnson, A. Menezes, and S. Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International Journal of Information Security*, 1(1):36–63, Aug 2001.

[32] A. Khalique, K. Singh, and S. Sood. Implementation of elliptic curve digital signature algorithm. *International Journal of Computer Applications*, 2(2):21–27, may 2010.

[33] J. Kishigami, S. Fujimura, H. Watanabe, A. Nakadaira, and A. Akutsu. The blockchain-based digital content distribution system. In *2015 IEEE Fifth International Conference on Big Data and Cloud Computing*, pages 187–190, Aug 2015.

[34] J. Laarakkers. Powermatcher: matching energy supply and demand to expand smart energy potential. https://www.tno.nl/media/1986/tno-powermatcher-jrv140416-01.pdf, 2007. Accessed May 2, 2018.

[35] H. Li, R. Lu, L. Zhou, B. Yang, and X. Shen. An efficient merkle-tree-based authentication scheme for smart grid. *IEEE Systems Journal*, 8(2):655–663, June 2014.

[36] S. Lindhurst. An analysis of shanks's algorithm for computing square roots in finite fields. In *Fifth Conference of the Canadian Number Theory Association*, June 1997.

[37] H. Lund. Renewable energy strategies for sustainable development. *Energy*, 32(6):912 – 919, 2007. Third Dubrovnik Conference on Sustainable Development of Energy, Water and Environment Systems.

[38] Magnr. Private versus public blockchains: Is there room for both to prevail? https://magnr.com/blog/technology/private-vs-public-blockchains-bitcoin/, 2016. Accessed June 25, 2018.

[39] A. Mehrizi-Sani and R. Iravani. Potential-function based control of a microgrid in islanded and grid-connected modes. *IEEE Transactions on Power Systems*, 25(4):1883–1891, Nov 2010.

[40] M. Meinshausen et al. Greenhouse-gas emission targets for limiting global warming to 2°c. *Nature*, 458:1158–62, 05 2009.

[41] K. Menyah and Y. Wolde-Rufael. Co2 emissions, nuclear energy, renewable energy and economic growth in the us. *Energy Policy*, 38(6):2911 – 2915, 2010. The Role of Trust in Managing Uncertainties in the Transition to a Sustainable Energy Economy, Special Section with Regular Papers.

[42] R.C. Merkle. Protocols for public key cryptosystems. In *1980 IEEE Symposium on Security and Privacy*, pages 122–122, April 1980.

[43] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck. Blockchain. *Business Information Systems Engineering*, 59(3):183–187, 2017.

[44] A. Oudalov, R. Cherkaoui, and A. Beguin. Sizing and optimal operation of battery energy storage system for peak shaving application. In *2007 IEEE Lausanne Power Tech*, pages 621–625, July 2007.

[45] P. Palensky and D. Dietrich. Demand side management: Demand response, intelligent energy systems, and smart loads. *IEEE Transactions on Industrial Informatics*, 7(3):381–388, Aug 2011.

[46] S. Raval. *Decentralized Applications: Harnessing Bitcoin's Blockchain Technology*. O'Reilly, 2016.

[47] P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In B. Roy and W. Meier, editors, *Fast Software Encryption*, pages 371–388, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[48] E. Rykwalder. The math behind bitcoin. `https://www.coindesk.com/math-behind-bitcoin/`, 2014. Accessed June 19, 2018.

[49] Blockchain Luxembourg S.A. Bitcoin block 529179. `https://blockchain.info/block/00000000000000000010e8c29888aa3f8607b4ad1317422acca8e04a2cd317d7`, 2017. Accessed June 25 2018.

[50] J. Salpakari, T. Rasku, J. Lindgren, and P.D. Lund. Flexibility of electric vehicles and space heating in net zero energy houses: an optimal control model with thermal dynamics and battery degradation. *Applied Energy*, 190:800 – 812, 2017.

[51] R.E.H. Sims, H. Rogner, and K. Gregory. Carbon emission and mitigation cost comparisons between fossil fuel, nuclear and renewable energy resources for electricity generation. *Energy Policy*, 31(13):1315 – 1326, 2003.

[52] N. Singh. Every disadvantage has its advantage: Reviewing blockchain. `https://hackernoon.com/`

every-disadvantage-has-its-advantage-reviewing-blockchain-1965e3462ba4, 2018. Accessed June 26, 2018.

[53] K. Smith. Vitalik — ethereum en route to a million transactions per second. https://bravenewcoin.com/news/vitalik-ethereum-en-route-to-a-million-transactions-per-second/, 2018. Accessed June 26, 2018.

[54] F.J. Solis and R.J.-B. Wets. Minimization by random search techniques. *Mathematics of Operations Research*, 6(1):19–30, 1981.

[55] J. Song. Blockchain: Separating hype from substance. https://eng.paxos.com/blockchain-separating-hype-from-substance, 2017. Accessed June 24, 2018.

[56] Team Sustain. Energy storage. http://www.teamsustain.com/energy-storage, 2017. Accessed June 27, 2018.

[57] M. Swan. *Blockchain: Blueprint for a New Economy*. O'Reilly, 2015.

[58] H.A. Toersche. *Effective and efficient coordination of flexibility in smart grids*. University of Twente., 2016.

[59] O. van Pruissen and R. Kamphuis. High concentration of heat pumps in suburban areas and reduction of their impact on the electricity network. *2011 IEEE Trondheim PowerTech*, pages 1–6, June 2011.

[60] S. Vaudenay. The security of dsa and ecdsa. In Y.G. Desmedt, editor, *Public Key Cryptography — PKC 2003*, pages 309–323, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[61] G. Verbong and F. Geels. The ongoing energy transition: Lessons from a socio-technical, multi-level analysis of the dutch electricity system (1960–2004). *Energy Policy*, 35(2):1025 – 1037, 2007.

[62] Bitcoin Wiki. Difficulty. https://en.bitcoin.it/wiki/Difficulty, 2017. Accessed June 24, 2018.

[63] Bitcoin Wiki. Block size limit controversy. https://en.bitcoin.it/wiki/Block_size_limit_controversy, 2018. Accessed June 26, 2018.

[64] Bitcoin Wiki. Mining. https://en.bitcoin.it/wiki/Mining, 2018. Accessed June 26, 2018.

[65] Bitcoin Wiki. Transaction fees. https://en.bitcoin.it/wiki/Transaction_fees, 2018. Accessed June 29, 2018.

[66] G. Zyskind, O. Nathan, and A. Pentland. Decentralizing privacy: Using blockchain to protect personal data. In *2015 IEEE Security and Privacy Workshops*, pages 180–184, May 2015.

A Python Files: Implementation Blockchain

This section provides a slightly simplified example of a blockchain implemented in Python.

```
import math
import hashlib
import time
import tkinter as tk


header_target = 0x1b0404cb
target = (int(str(header_target)[2:8], 16)*2**(8*(int(str
    (header_target)[0:2], 16)-3)))

version_number = '00000001'

transaction_pool = []

version_list = []
previoushash_list = []
merklehash_list = []
time_list = []
targethash_list = []
nonce_list = []

hashlist_blocks = []

# SHA-256 hash a string
def encrypt(string):
    return hashlib.sha256(string.encode()).hexdigest()

# Check whether a given hash is valid (that is if the
    hash value is lower than the target
def valid_hash(hash):
    to_check = int(hash, 16)
    if to_check < target:
        return True
    else:
        return False

# Returns the hash of the previous block
def last_hash():
    return hashlist_blocks[len(hashlist_blocks) - 1]
```

```
# Determines the merkle root hash of all transactions of
    the block
def merkle_root(stringlist):
    hashed_list = []
    for i in range(0, len(stringlist)):
        hashed_list.append(encrypt(stringlist[i]))
    current_level = hashed_list
    next_level = []
    while len(current_level) > 1:
        if len(current_level) % 2 == 0:
            for i in range(0, math.ceil(len(current_level
                )/2)):
                next_level.append(encrypt(current_level
                    [2*i] + current_level[2*i+1]))
            current_level = next_level
            next_level = []
        else:
            for i in range(0, math.ceil(len(current_level
                )/2)-1):
                next_level.append(encrypt(current_level
                    [2*i] + current_level[2*i+1]))
            next_level.append(encrypt(current_level[len(
                current_level)-1] + current_level[len(
                current_level)-1]))
            current_level = next_level
            next_level = []
    return current_level

# Hashing a block
def hash_block(version, previous_hash, root_hash,
    timestamp, target):
    hash = '0'
    nonce = 0
    while valid_hash(hash) == False:
        header = version + previous_hash + root_hash +
            timestamp + target + '0' * (8 - len(hex(nonce)
            )) + hex(nonce)
        hash = encrypt(header)
        nonce += 1
        # print(nonce)
    hashlist_blocks.append(hash)
    version_list.append(version)
    previoushash_list.append(previous_hash)
    merklehash_list.append(root_hash)
    time_list.append(timestamp)
    targethash_list.append(target)
```

```
    nonce_list.append(nonce)

# Define the first block, the merkle root is currently a
    random string
def genesis_block():
    version = version_number
    previous_hash = '0' * 64
    merkle_root = encrypt("Hello, I am the first block")
    timestamp = hex(round(time.time()))
    hash_block(version, previous_hash, merkle_root,
        timestamp, target)

# Takes the transactions from the transaction pool, and
    creates and adds the block to the chain
def add_block():
    if len(transaction_pool) > 0:
        data = transaction_pool
        version = version_number
        previous_hash = last_hash()
        root_hash = merkle_root(data)
        timestamp = hex(round(time.time()))
        hash_block(version, previous_hash, root_hash,
            timestamp, target)
    else:
        print("There are no transactions to be stored")

# A print statement that shows what is stored in the
    blockchain
def show_all_blocks():
    for i in range(0, len(hashlist_blocks)):
        print('All data stored in block', i, ':')
        print('Version :', version_list[i])
        print('Hash of previous block :',
            previoushash_list[i])
        print('Merkle root hash of transactions :',
            merklehash_list[i])
        print('Time at when the block is created :',
            time_list[i])
        print('Target hash :', targethash_list[i])
        print('Nonce :', nonce_list[i])
        print('Hash of this block :', hashlist_blocks[i])
        print('\n')

sup_from = None
sup_to = None
sup_amount = None
```

```
filled = None

# An interface for the person that gives energy (let's
    call him the supplier)
def PopUp_Supplier():
    window = tk.Tk()
    window.title("Information of Supplier")

    Label1 = tk.Label(window, text="Name:", font=("
        Cambria", 14))
    Label1.grid(row=1)
    textentry1 = tk.Entry(window, bd=5, font=("Cambria",
        14))
    textentry1.grid(row=1, column=1)

    Label2 = tk.Label(window, text="To:", font=("Cambria
        ", 14))
    Label2.grid(row=2)
    textentry2 = tk.Entry(window, bd=5, font=("Cambria",
        14))
    textentry2.grid(row=2, column=1)

    Label3 = tk.Label(window, text="Amount (kWh):", font
        =("Cambria", 14))
    Label3.grid(row=3)
    textentry3 = tk.Entry(window, bd=5, font=("Cambria",
        14))
    textentry3.grid(row=3, column=1)

    def sup_getEntry():
        global sup_from, sup_to, sup_amount, filled
        sup_from = textentry1.get()
        sup_to = textentry2.get()
        sup_amount = textentry3.get()
        filled = True
        window.destroy()

    def quit():
        global filled
        filled = False
        window.destroy()

    b1 = tk.Button(window, text="Ok", command=
        sup_getEntry, font=("Cambria", 14)).grid(row=4,
        column=0)
```

```
    b2 = tk.Button(window, text="Cancel", command=quit,
        font=("Cambria", 14)).grid(row=4, column=1)

    window.mainloop()

    if filled == True:
        PopUp_Receiver()

rec_confirm = None

# An interface for the person that receives the energy
def PopUp_Receiver():
    window = tk.Tk()
    window.title("Confirmation Transaction")

    label1 = tk.Label(window, text="Name: "+str(sup_to),
        font=("Cambria", 14))
    label1.grid(row=1, column=1)

    label2 = tk.Label(window, text="Transaction: "+str(
        sup_amount)+" kWh from "+str(sup_from), font=("
        Cambria", 14))
    label2.grid(row=2, column=1)

    def confirmed():
        global rec_confirm
        rec_confirm = True
        transaction_pool.append(sup_from + " sends " +
            sup_amount + " kWh to " + sup_to)
        window.destroy()

    def denied():
        global rec_confirm
        rec_confirm = False
        window.destroy()

    B1 = tk.Button(window, text="Yes", command=confirmed,
        bg="green", fg="white", font=("Cambria", 14))
    B1.grid(row=3, column=0)
    B2 = tk.Button(window, text="No", command=denied, bg
        ="red", fg="white", font=("Cambria", 14))
    B2.grid(row=3, column=2)
    window.mainloop()

    print("rec_confirm =", rec_confirm)
```

```
    if rec_confirm == True: PopUp_Confirmed()
    elif rec_confirm == False: PopUp_Denied()

# A pop-up for the supplier if the receiver confirms the
    transaction
def PopUp_Confirmed():
    window = tk.Tk()
    window.title("Transaction Confirmed")

    label1 = tk.Label(window, text="Your transaction:",
        font=("Cambria", 14))
    label1.grid(row=0, column=0)
    label2 = tk.Label(window, text=sup_amount+" kWh from
        "+sup_from+" to "+sup_to, font=("Cambria", 14))
    label2.grid(row=1, column=0)
    label3 = tk.Label(window, text="has been confirmed.",
         font=("Cambria", 14))
    label3.grid(row=2, column=0)

    def okay():
        window.destroy()

    B1 = tk.Button(window, text="Ok", command=okay, font
        =("Cambria", 14)).grid(row=3)

    window.mainloop()

# A pop-up for the supplier if the receiver denies the
    transaction
def PopUp_Denied():
    window = tk.Tk()
    window.title("Transaction Confirmed")

    label1 = tk.Label(window, text="Your transaction:",
        font=("Cambria", 14))
    label1.grid(row=0, column=0)
    label2 = tk.Label(window, text=sup_amount + " kWh
        from" + sup_from + " to " + sup_to, font=("Cambria
        ", 14))
    label2.grid(row=1, column=0)
    label3 = tk.Label(window, text="has been denied.",
        font=("Cambria", 14))
    label3.grid(row=2, column=0)

    def okay():
        window.destroy()
```

```
    B1 = tk.Button(window, text="Ok", command=okay, font
        =("Cambria", 14)).grid(row=3)

    window.mainloop()


# An example of how creating this blockchain would look
    like

# Create the first block
genesis_block()

# Ask the supplier what the transaction is, ask the
    receiver if this is correct, show the supplier
    confirmation/denial
PopUp_Supplier()

# Add all current transactions in the transaction pool to
     a new block and add this to the chain
add_block()

# Show what is stored in the blockchain
show_all_blocks()
```