StromSack - An ElectriciTeam Project

Demo Website
https://electriciteam.github.io/demo/

The software code which is part of this report is open source and available at: https://github.com/electriciteam

Liliane Ableitner lableitner@ethz.ch
Ananya Joshi joshia@ethz.ch
Marti Perez Canyelles martipe@student.ethz.ch
Adrian Sarbach sarbacad@student.ethz.ch
Sandro Schopfer sandro.schopfer@ethz.ch
Arne Wulff awulff@student.ethz.ch
Li-Wei Yap yapl@student.ethz.ch

ETH Zürich, 30.04.2018

All members contributed equally to this report

1. Introduction

This project report was written as part of the week-long spring 2018 course 'Blockchain And the Internet of Things (851-0591-01L)' run by Professors M. Dapp, S. Klauser, and D. Helbing. In the scope of this course, we chose EWZ's (the local utility company of Zurich) challenge *Virtual Energy Storage*. Our task was to create a blockchain solution to find possible answers for the question, "How might blockchain technology help to develop a reliable and highly efficient virtual energy solution that gives anyone the possibility to produce, store and use his/her own energy anywhere and anytime?". This means that each prosumer (producer and consumer) can store their energy in a virtual account, similar to a bank account. As soon as the prosumer needs energy, he/she can access the energy on the virtual bank for personal consumption. Another requirement of the challenge was that the prosumers should be able to use their energy anywhere, i.e. outside of their homes and out of reach of personal smart meters. This document gives an overview of the obtained results, including a description of the general concept, the solution design, testing results and a future outlook.

Our team's (ElectriciTeam) solution, called *StromSack*, aims to enable 100% green, home-made electricity, to be consumed anywhere and anytime. This is done by tracking production of energy (from any solar photovoltaic (PV) system or wind turbine one may have at home), and consumption of energy at home or elsewhere. An IoT device (*Raspberry Pi*) allows tracking the production and consumption, using EmonPi (a Raspberry Pi Energy Monitoring Shield) to access the prosumer data. This is backed up by an Ethereum Smart Contract `Contract.sol`, which tracks transactions and ensures that they are safe. For further reference, the address of the contract is `0x0710f1f272ffa3219952ce599ee6c08dc6c9ad53`, and it is located in the Ropsten Test Network.

Short overview of blockchains

The blockchain technology grew out of innovations in the cryptocurrency Bitcoin, and it is also known as the distributed ledger technology. While most systems keep a single, centralised copy of the transactions and balances across a network, blockchains allow the users of the network to record and share an identical copy of these transactions in a decentralised fashion. Due to the massive redundancy of copies of the ledger of transactions, it is difficult to tamper with the transactional data, thus safeguarding data validity. Trust amongst users is further fostered by the use of cryptographic mechanisms. Cryptocurrency miners have to solve a mathematical hash problem, by randomly trying numbers, in order to build blocks (i.e. record transactions that have previously just occurred) and extend the blockchain. The first miner to solve the problem broadcasts the newly formed block to the network, and other miners validate the block. Because of the high computational intensity of mining, it would be too costly to tamper with a blockchain and modify the history of transactions and balances, thus further improving the security of the blockchain. The cryptocurrency Ethereum has further contributed to blockchain technology with the concept of smart contracts, which are computer codes that help to verify or enforce the performance of the transactions. Smart contracts are executed upon the initiation of transactions or by Oracles, which are software agents that find and verify real-world conditions that initiate transactions themselves and submit this information to the blockchain **[9]**. As will be explained in the following section, smart contracts require Decentralised Applications (DApp) to work.

2. Concept

Problem statement and presentation of solution

The Swiss Energy Strategy 2050 aims to promote the use of renewable energies within Switzerland, and to ensure a long-term energy supply regardless of economic, political, and technological developments both in Switzerland and abroad **[1]**. The decreasing costs of renewable energy sources have fostered a boost in adoption of solar and wind energy **[2]**. This movement is driven not only by large energy companies, but also by the consumers who contribute by installing PV systems or small wind turbines on their own land and producing their own energy. We call these households *prosumers* **[3]**. Besides initial investment costs, installation, and maintenance of the technology, prosumers who own the more popular PV systems (compared to wind turbines) face yet another challenge, namely their asynchronous consumption and production profiles: PV systems harvest energy during the day while the sun is shining, whereas the household's major demand of electricity is during morning and evening time when PV production is low. This normally allows households to self-consume only approximately 30% of their own production **[4]**; the surplus energy that is not immediately consumed needs to be stored with the help of batteries or fed into a public energy grid. As the first option comes with a high initial investment cost that not everyone is willing to bear **[5]**, the second option is more feasible for a majority of prosumers. That being said, feed-in tariffs are quite low and prosumers need to buy back electricity at a high price from the utility during times of the day when needed. This situation causes many consumers to rethink their transformation towards being a prosumer **[6]** and so far, the electricity grid in Switzerland remains a one-sided affair with energy being supplied by centralized utilities to industrial and residential consumers.

In order to support the renewable energy production movement, or more precisely the transformation from consumers to prosumers, we introduce StromSack - a decentralized application (DApp) that allows prosumers to maximize their self-consumption without large investments in physical batteries. StromSack is a virtual energy storage inspired by a solution already applied in Germany **[7]** and is built on blockchain technology. Where integrated utilities once controlled the entire system, StromSack would support an electricity grid in which energy production, storage, and consumption are all decentralized. StromSack enables prosumers to trade their surplus energy for the ElectricityCoin (ELC) token and, in turn, use these tokens when buying electricity - either from the utility or any other paid energy service like electric vehicle (EV) charging. With this service, we promote the prosumer's autonomy and hope to ensure the utility's competitive advantage for the scenario of liberal markets in the future.

Functionality of StromSack and Use Cases

StromSack is a DApp building on a smart contract which is owned by the utility company (which also operates the local distribution network and supplies the area with electricity). In the energy sector, high hopes have been placed on the blockchain as an enabling technology **[8, 9]**: it allows the exact tracking of energy flows in networks and stores all the transactions in a trustful and secure manner. Due to its decentralized nature, data validity is ensured and data fraud is hardly possible. Many different blockchain technologies exist to date, but many others are still in their infancy. Given

Ethereum's maturity, we decided to build our application on the Ethereum network and create our own ERC20 token.

In order to benefit from the StromSack programme, a household or an institution first needs to have a form of renewable energy source, e.g. a PV system or a wind turbine which gives them the status of a prosumer **[3]**. Both electricity production and consumption are measured by smart meters which are able to communicate with a smart contract on a blockchain. On the financial side, a prosumer needs an Ethereum wallet address for receiving and spending ELC. For the end user, StromSack provides a user interface that gives insights into current ELC balance, energy production and consumption and transaction history.

Any external energy service provider (e.g. an EV charging station) can join the programme by installing the required smart meters (for measuring the customer's energy consumption) and by creating an Ethereum wallet address (to receive ELC as a payment method). In the future, it is conceivable that blockchain-enabled smart meters will be available - for the time being, we have to separate these two components.

Figure 1 depicts all interactions between StromSack entities on the three dimensions: token flow, energy flow, and communication.

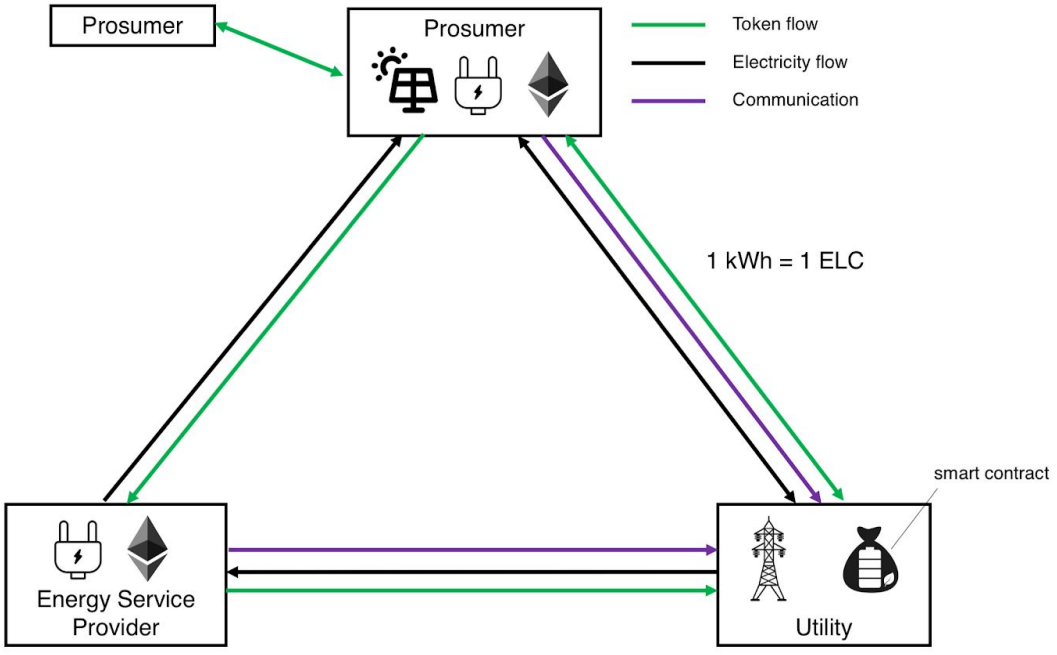
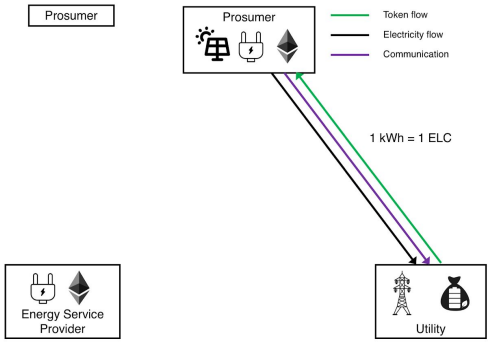


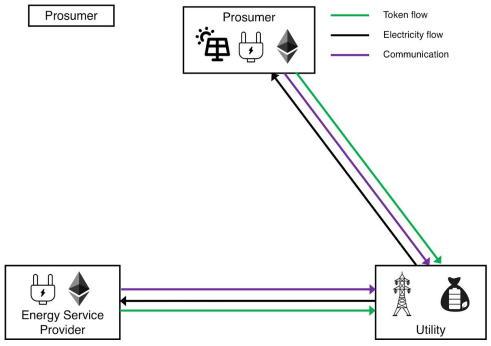
Figure 1: *Token flows, energy flows, and communication within StromSack network*

Use Case: Exchanging surplus electricity for ELC tokens



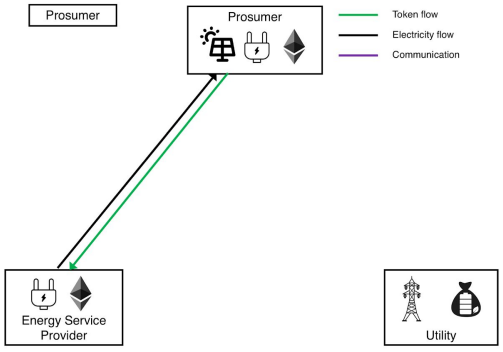
Prosumers can feed their surplus energy (i.e. the energy which is not immediately consumed) into the local distribution grid operated by a utility company. The smart meter at home measures the energy flow and triggers an event in the smart contract, owned by the utility, by communicating the amount of energy fed in. In turn, the smart contract issues the tokens at an exchange rate of 1kWh = 1ELC to the prosumer's wallet address.

Use Case: Buying electricity from the grid using ELC tokens



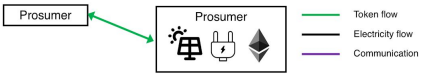
Whenever needed, the prosumer or an energy service provider can buy electricity with ELC from the utility. The smart meter registers an incoming energy flow, reports it to the smart contract, and triggers the payment with ELC (after subtracting a service fee). If required, the prosumer can request a second smart meter for measuring own consumption at another location (e.g. a holiday home).

Use Case: Charging personal devices at charging stations



Prosumers have the possibility to use any paid electricity service outside their home (e.g. outside their own consumption area which is equipped with a smart meter). An external service (e.g. an EV charging station) supplies the prosumer with electricity and accepts ELC as payment method. **Figure 2** provides the reader with more information on charging devices outside the home.

Use Case: Exchanging ELC tokens with others



StromSack customers, moreover, have the possibility to trade ELC or to make ELC gifts to peers. With ELC being a ERC20 token, it will be automatically available for exchange.

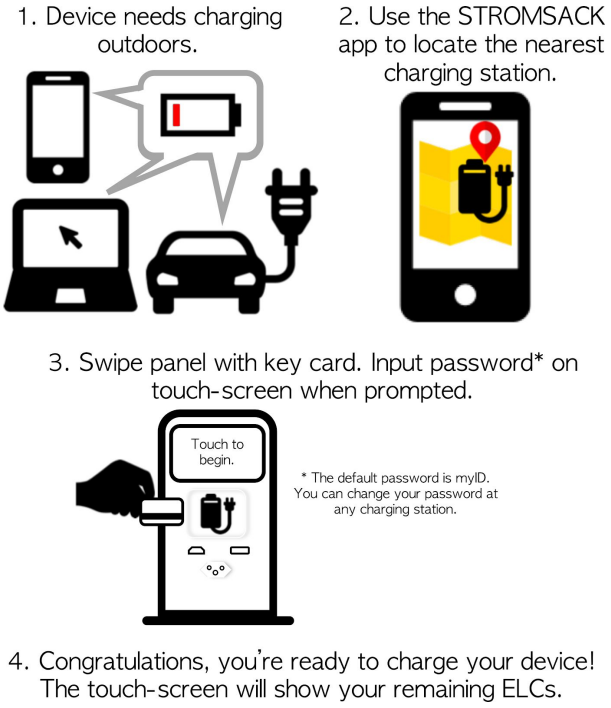


Figure 2: *Use case explanation for the customer*

3. Solution Design

Prior to BIOTS, very few members of the team had exposure to blockchain technology. To design our solution, we came up with a variety of ideas. Some early designs included using a smart plug to measure electricity use in public spaces, or mandating public spaces to install smart plugs. Although these solutions could have been implemented with blockchain, human behaviour would render these solutions useless. This is because if someone could get power for free in public spaces, such as at work or at school, then there would be little incentive to use a smart charger where one would have to pay. Thus, we settled on the idea of an energy market focused on large electrical stations that consumers are already paying for. We developed our concept a bit further with other future interesting applications which essentially use the same basic principles, like charging electric cars, or backup power.

Our solution consists of the smart contract, a website, and a hardware IoT sensor. We will explore these three components and their interactions in detail in the following paragraphs.

Smart Contract

The first step was to develop the smart contract. We adopted the smart contract presented by Elyssium. We make use of the `SafeMath.sol` and modified `Token.sol` to create the contract for the ELC token, which is based on the ERC20 standard. As we learnt, we need `SafeMath` to prevent overflow and allow for safe addition and subtraction. We used the Remix Ethereum IDE to run our solidity code.

```
using SafeMath for uint256;
address owner; //=0x14723a09acff6d2a60dcdf7aa4aff308fddc160c
uint256 totalSupply = 10000;
uint256 tokensLeft = 10000;
uint256 serviceFee = 0 ;
```

We set the address owner as a global variable. This ideally is the utility company, so they can moderate the transactions. We keep a record of the tokens left (tokens not in circulation) and added a position for a static service fee. In the end, we attempted a dynamic service fee as a percentage of the ELC value being transferred. The basic features of the contract were the deposit and withdraw functions. Using the safe math module, we ensured these basic functions work. We also added methods to check, calculate, and transfer service fees. In solidity, these features were functional.

The most challenging step was implementing the service fees. Currently, prosumers can sell their additional energy to the grid as a fixed rate of 8Rp/kWh. They can buy back that energy for approximately 26Rp/kWh. We recognize that utility companies needs to make a profit off of this virtual storage system in the future. From our point of view, a reasonable fee is 15 szabo/kWh (1 million szabo = 1 Ether), independent of gas fees, directly to the utility company. However, as we discovered during BIOTS, there is a difference between transferring ELC and transferring Ether. Furthermore, it is an even bigger challenge to do both at once. Despite completing several tutorials, we were unable to properly implement this transaction fee.

One approach was to put the transaction in the smart contract, which is possible with a `require` statement. In Remix this solution worked; however, in our web3 implementation, the transaction fee was not subtracted from the user's account and did not go to the contract holder. Using a web3 object to transfer Ether did not work either. We tried many alternatives, including creating a new local ethereum wallet, removing certain lines from the smart contract, restarting the web client several times, and waiting for updates to take place. Debugging was very difficult and even after getting support from a few advisors, we couldn't figure out the issues. Although this aspect remained unimplemented in the final product, it is integral to our concept.

Website

The second component, the website, is organized in a user friendly manner. It is hosted on github and available on https://electriciteam.github.io/demo/index. Users can register for StromSack, deposit and withdraw ELC, gift ELC, and buy equipment to become prosumers **(Figure 3a)**. The website was created with HTML, Javascript, and CSS.

In our `custom.js` file, we combine the front and back-end of our website. First, we check if web3 has been injected by the browser. Then, we create an instance of our contract object, which we use to interact with the smart contract. Finally, we allow tokens to be withdrawn or deposited, while checking for errors **(Figure 3b)**. In the future, we would update our website to dynamically show changes in an account balance.

Hardware

On the hardware side, we use a Raspberry Pi equipped with sensors to measure the electric current from production. During the week of the hackathon, we actually had this hardware already set up, so we were able to check one prosumer's data from a small town close to Zurich through an API built in the Raspberry Pi. Based on the data analysis, it was clear that in Zurich, StromSack could work. Energy production, even in February could account for a large part of energy consumption. If a household were energy efficient, there is a high chance there is an energy surplus. This data, together with household electricity consumption information, influences the decision on electricity pricing.

The Raspberry Pi **(Figure 3c)** is a feasible smart meter and sensor to use because they are both ubiquitous and relatively inexpensive, at projected costs of approximately €200 per unit. Ideally, the hardware is available to every prosumer. Any electricity the prosumer has produced but not used immediately is stored in the account, as shown in the section 2. Through the Raspberry Pi and the smart contract, the energy surplus is automatically converted into ELC coins and credited to the user's account. Upon use of electricity, the utility charges a service fee and the user pays for the consumed electricity with ELC.

The flexibility of this hardware would also be useful when updating the system: instead of changing all the hardware, we could just update the device "over-the-air" (OTA). As a limitation, people could argue that it is an easily hackable device, but that could be mitigated by a strong software that could detect modifications either in the hardware or software, automatically raising an alert to the utility company. Then, an OTA update that secures the system again could be rolled out quickly.

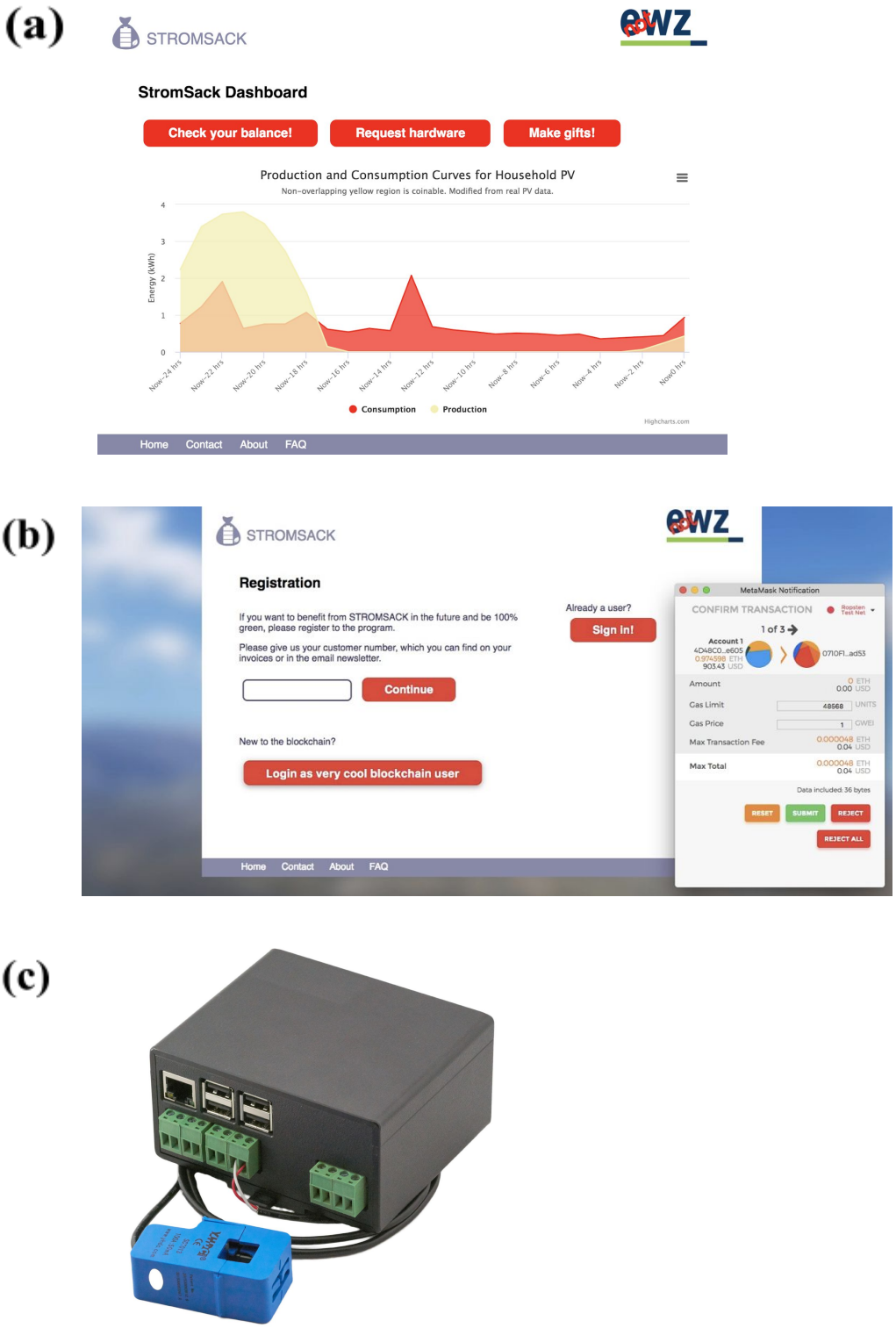


Figure 3: (a) *The dashboard is loaded immediately upon logging into one's account. It allows the user to track his energy production and consumption in the past 24 hours. There are also options to request for additional smart meters and to deposit, withdraw, or trade ELC.* ***(b)*** *Smart contract interacting with web3 metamask with a real ERC20 token.* ***(c)*** *Raspberry Pi.*

4. Evaluation and Testing

Our working prototype **(Figure 4a)** is a scalable solution that acts like a bank account for energy: when you produce energy, you get ELC tokens, whereas when you consume energy, you spend them. We assume the production always takes place at one's home, where one has a fully working smart meter for tracking both production and consumption (Raspberry Pi prototype). This smart meter was designed in a previous project of one of our team members. The energy, however, can be consumed at any "energy ATM", meaning that one would need just a StromSack identification card for the "energy ATM" to recognise him and charge his account.

Front end

Diving into more detail, our solution consists of a front end where the user can check his account, buy/sell tokens, check their production and consumption, trade tokens, request the needed hardware (smart meters for home, additional cards for using at the energy ATMs, or any other item related with the service).

The login to the front end uses one's personal Metamask account, which is also used for storing one's ELC tokens. For demonstration purposes, we set up an initial amount of 5000 ELC. This needs to be connected to the data coming from the energy meter (Raspberry Pi or "energy ATM"), so that it is updated automatically according to one's production and consumption.

An error found in the front-end solution is the need to force refresh the page, including deleting all caches, in order to see one's updated balance. This is both for deposit and withdrawal.

A further change to the implementation that should be made is the histogram, which has been implemented using Highcharts. For demonstration purposes, all data is hard-coded. Ideally, this data comes from the Raspberry Pi that tracks the consumption and production at home, combined with the data coming from the consumption of energy away from home, at the "energy ATMs". Due to the time-constraint during the project, we could not implement this.

On a side note, it is important to mention that further improvements on the user interface should be implemented: A more friendly user interface, with explanations that make the user understand how to use the product should be developed. For demonstration purposes, and considering the time-constraint, we designed only a minimalist user interface.

Smart Contract

From our point of view, the smart contract owned by the utility company automatically interacts with the blockchain whenever you consume or produce energy. Our initial idea was to implement a fee for energy transfers. We achieved our goal while programming on Remix (Solidity IDE), but once we tried to communicate with the smart contract with the front-end using injected web3, the transaction failed. Thus, the tokens were not transferred and the fee was not paid. We contacted one of the blockchain experts in BIOTS, but even with help, the problem remained unsolved. It is still unclear

why the implementation of the transaction fee fails. For demonstration purposes, we removed it from our code.

The following code is part of our Solidity contract with transaction fee:

```
function withdrawEnergy(uint256 amountEnergy) public payable{
    serviceFee = amountEnergy*1 ether;
    require(msg.value >= serviceFee);
    owner.transfer(serviceFee);
    //token
    require(balances[msg.sender]>amountEnergy);
    balances[msg.sender] -= amountEnergy;
    tokensLeft += amountEnergy;
}
function getServiceFee() public{
    transfer from contract to owner of contract
    require(msg.sender == owner);
    msg.sender.transfer(this.balance);
    owner.transfer(this.balance)
}
```

Security against cyber attacks

Absolute security does not exist. It is true that we can have as much security as we want, the question is *what are we willing to give up to get it*:

- **Confidentiality**: Prevention of unauthorized disclosure of information.
- **Integrity**: Prevention/detection of unauthorized modification or deletion of information.
- **Availability**: Prevention of unauthorized withholding of information or service.

Depending on the industry, one is going to be more critical than another: banks look for integrity, online retailers look for availability, the military looks for confidentiality, etc.

In our case, integrity of data is the most critical aspect, and that is the reason why we are using blockchain technology with smart contracts. It is true that the demonstration itself works, however, as we are not experts on coding smart contracts, we believe that more security tests should be done in order to ensure that the whole system is safe. There are consulting companies who offer these services, so we strongly advise to check the concept as a whole with them before any further development.

Distributed consumption

For distributed energy consumption, we consider energy to only be consumed at “energy ATMs” or at home. In the case of holiday homes, a normal smart meter will be installed there and connected to one’s account, so that everything is integrated into one single account. This account is shared with all household members.

Every household member will have a card, which can be used at any "energy ATM", from charging stations for electric vehicles to smaller charging stations placed at different locations (train stations, restaurants, parks, etc.) where one can charge any smaller electric devices, including one's phone, laptop, or other similar devices.

Other areas such schools or offices are not considered for this "energy ATM" solution, as people in these buildings are either paying to use the facilities, or generating revenue to the company located in the building. As a result, the energy used inside the buildings should be free for the user.

The integrated account **(Figure 4b)** is the metamask account, which is already working and fully capable of storing the tokens. However, further developments have to be made for the ATM system (measurements and communication between ATM and the integrated account), and the previously mentioned connection between the smart meters placed at the different residences and the integrated account. For demonstration purposes, all this has been considered under the `withdrawEnergy` function, which updates the number of tokens one's personal integrated account has left once a withdrawal has been completed. The withdrawn tokens are actually transferred from the user's account to the smart contract, which generally is owned by the utility company. During this transaction, the previously mentioned transaction fee has to be paid to the utility company, but as explained before, we were not able to implement it as it caused the transaction to fail.

Technical advantages of our solution

In comparison to other approaches to this challenge, our solution is practical, cheap, and disruptive in the sense that we offer one single decentralized platform that controls all the parts of the distributed grid automatically. There are other companies who have created energy markets that are similar to our solution. However, these companies have inflated prices and operate on a membership system, probably not using blockchain and not offering a single global solution, which is both transparent (and so, trustable) for the users who want to know more, but also simple to use for the everyday user.

Moreover, this kind of design can be further developed so that any company around the globe can take part in it, as the transaction fees are the only point of discussion between the different companies.

Future Challenges

The StromSack product will need to be adapted to enable near real-time transactions to balance the virtual storages with the actual grid. As transaction costs and environmental externalities increase with increased transaction throughput, smart-meter values will need to be signed off-chain. Aggregated meter-values for the net load (PV generated power minus load) can be committed to the chain in regular, but infrequent intervals (e.g. monthly) for settlement. The approach of signing state changes off-chain is referred to as *state channels*.

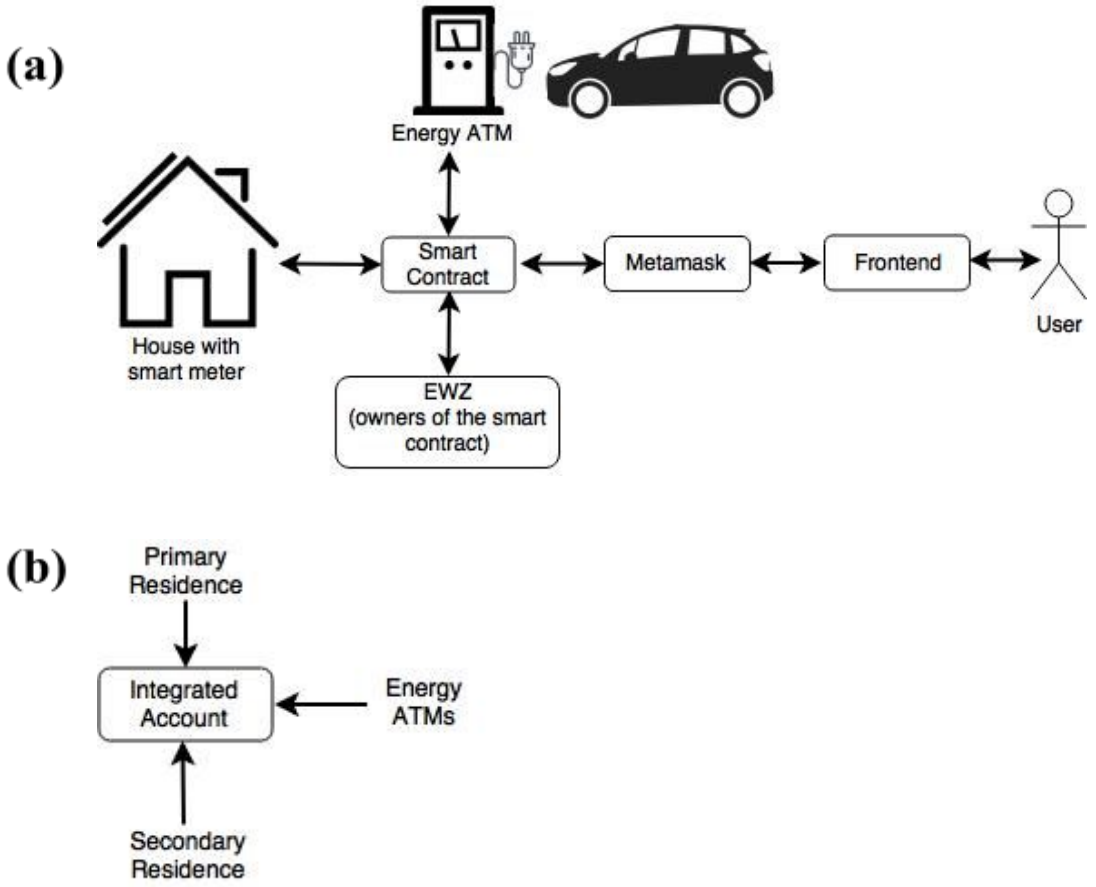


Figure 4: (a) *StromSack ecosystem.* ***(b)*** *Overview of the integrated account influences.*

5. Conclusion and Outlook

The imminent shift in energy production, as well as the change in electricity needs, from electric cars to mobile devices, and a growing awareness of the environment, forces energy companies to adapt. StromSack is a practical way for prosumers to reuse the energy in multiple places after it has been injected to the grid. Its simplicity and broad applicability, from sharing your energy to charging your car at a proprietary charging station, are the strengths of our solution. StromSack ensures green energy consumption, produced by yourself, anywhere and anytime. With a growing market in renewable energy production and storage, and an increasing number of users, StromSack will become increasingly useful. The implementation of tokens opens up the possibility to trade renewable energy in an easy way.

In conclusion, the prototype developed during the BIOTS2018 Challenge demonstrated the practicability of our solution and its main features. Our decentralized solution is not only secure, safe from cyber attacks and data fraud, but it is also practical and cost-efficient. Still, big developments remain to be accomplished on both the hardware and the software, as well as on its roll-out to houses and charging stations. Also, several governmental regulations would have to be overcome, in order to make StromSack even more advantageous on a large scale. In the scope of our project, further work would include the implementation of the transaction fees for energy transfers in conjunction with the smart contract, as well as the development of a more user-friendly interface on the StromSack website and the implementation of off-chain transactions to avoid high gas costs. Additionally, when the former is achieved, StromSack could be further adapted to enable near real-time transactions. Nevertheless, we strongly believe that an implementation of this project would be a success: if classical utility companies do not implement it, another energy service provider might do so faster, leaving the classical companies out of business.

6. References

[1]: Swiss Federal Office of Energy SFOE. (2018). *Energy Strategy 2050.* Available from: http://www.bfe.admin.ch/energiestrategie2050/index.html?lang=en

[2]: Carrington, D. (2017). *'Spectacular' drop in renewable energy costs leads to record global boost.* Available from: https://www.theguardian.com/environment/2017/jun/06/spectacular-drop-in-renewable-energy-costs-leads-to-record-global-boost

[3] Morstyn, T., Farrell, N., Darby, S. J., & McCulloch, M. D. (2018). Using peer-to-peer energy-trading platforms to incentivize prosumers to form federated power plants. *Nature Energy*, *3*(2), 94.

[4] Luthander, R., Widén, J., Nilsson, D., & Palm, J. (2015). Photovoltaic self-consumption in buildings: A review. *Applied Energy*, *142*, 80-94.

[5] Devine-Wright, P., Batel, S., Aas, O., Sovacool, B., Labelle, M. C., & Ruud, A. (2017). A conceptual framework for understanding the social acceptance of energy infrastructure: Insights from energy storage. *Energy Policy*, *107*, 27-31.

[6] Palm, J. (2018). Household installation of solar panels–Motives and barriers in a 10-year perspective. *Energy Policy*, *113*, 1-8.

[7]: E.ON SolarCloud - Ihr virtueller Stromspeicher. Available from: https://www.eon-solar.de/eon-solarcloud

[8] Basden, J., & Cottrell, M. (2017). How utilities are using blockchain to modernize the grid. *Harvard Business Review*.

[9]: Meunier, S. *Blockchain 101: What is Blockchain and how does this revolutionary technology work?*

7. Licence