symBIOTic

Roman Blum (romblum@student.ethz.ch)
Petar Ivanov (ivanovpe@student.ethz.ch)
Andrei Isac (isaca@student.ethz.ch)
Anne-Sophie Mertgen (amertgen@student.ethz.ch)
Jan-Philipp Schulze (jschulze@student.ethz.ch)
Andrei Ursache (ura@student.ethz.ch)
Cameron Weibel (caweibel@student.ethz.ch)

ETH Zürich, 04/30/2018
This report is licensed under the CreativeCommons licence CC BY-SA v4.0.

This project report was written as part of the spring 2018 course “Blockchain And the Internet of Things (851-0591-01L)” run by M. Dapp, S. Klauser, and D. Helbing.

This report is licensed under a CreativeCommons licence CC BY-SA v4.0

The software code which is part of this report is open source and available at https://github.com/ETHBiots2018/symBIOTic .

Disclaimer. The work for this project has been evenly split between our team members, some of us worked more on the implementation, while some more on the documentation.

Table of Contents

What is the problem? (CW, ASM)

Waste management is an issue of global importance, both figuratively and literally. As the population continues to grow and the level of consumerism rises, humans continue to produce more waste.[1] The problem of how to manage that waste and ensure the minimization of damage to environment is of paramount importance to the survival of our biosphere. One of the most popular methods proposed to deal with waste generation is recycling: disposing of the waste in a way that the material can be processed and reused. Although in theory responsible recycling can dramatically reduce the global waste, in practice only 9% of the 6.3 billion metric tons plastic waste was recycled globally in 2017(around 30% in the USA, around 50% in the EU in 2016). [1–3] Therein lies the challenge. How do we incentivize consumers to recycle their waste?

Some countries, such as Germany, Sweden, and the US already have systems established how to incentify the return or plastic bottles to the retailers by a monetary deposit charged for every bottle purchased. Upon return of the bottle, the consumer obtains the deposit from a collection machine.[3] These systems, combined with stricter regulations and awareness of recycling, have led to a bottle-recycling rate of 93% in Germany, 83% in Sweden and 37% in the US; all well above the global average. The difference in the efficiency of the incentive appears to be related to the amount of money received for each bottle. While in the US you only get USD 0.05 (EUR 0.04), in Sweden you get SEK 1-2 (EUR 0.10-0.20) in Germany you get EUR 0.25 returned for each PET bottle.[4] This suggests that more generous rewards for the waste would increase the consumers' motivation to return their plastic bottles. The success of such incentivisation systems, however, depends very much on the financial motivation of the consumers' demographics and financial motivations. Deposits of a few cents (5 or 25 ct.) may not motivate high-income groups as much as low income groups.[5]

Our Solution

Overview (CW, ASM)

To provide financial incentive for recycling, we have built upon the existing recycling reward systems already active in countries like the US and Germany, where a consumer can deposit different kinds of waste and receive a monetary reward from a, to put it simply, a reverse vending machine. We use the blockchain to allow users to recycle different kinds of waste and receive not only immediate monetary value like one would receive at one of the existing systems, but also a number of tokens to allow the consumers to take part in a lottery. Anybody with possession of these tokens has the opportunity to win a large sum of money with a probability proportional to the number of tokens they have amassed from their recycling/trading efforts. We mention trading as there would likely be a secondary market for the buying and

selling of these tokens, where an owner of the token could have gained his share through trading rather than recycling.

How does it work? (CW, JPS)

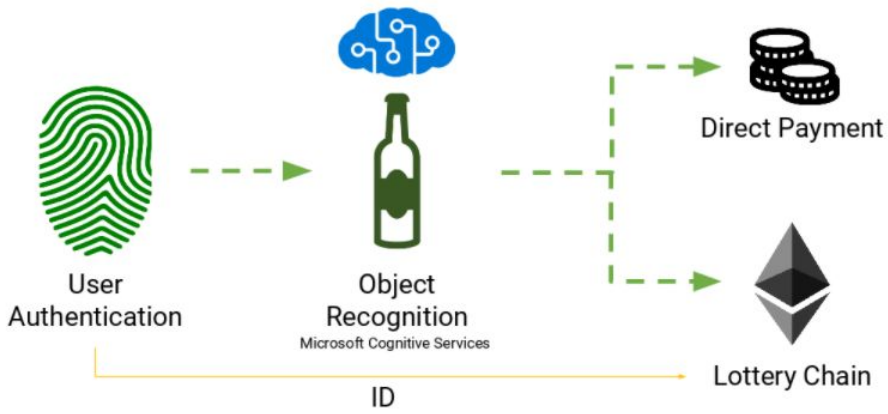

Figure 1: Overview of the process

From a consumer perspective the system works as follows:

1. Visit one of our machines.
2. Scan your finger to login/register with our system.
3. Deposit any number of PET bottles and/or other supported waste.
4. Your account will be credited with both your local currency and ANSO based on the quantity and type of recycled waste.
5. You can visit a collection point to withdraw money or use open API to trade the ANSO.
6. As soon as a certain amount of tokens was generated (this is a fixed amount each draw of the lottery), you will have the chance, based on the number of tokens you have earned that week, to win a large sum of money.

From an operator's perspective the system can be implemented on top of their existing recycling solution. For example, let's imagine you own a grocery store in Dusseldorf and you want to implement our solution. The system, from the operator's perspective works as follows:

1. Consumers deposit PET bottles and/or other supported waste.
2. You as an operator receive a certain amount of money for each PET bottle (deposit, let's say €0.30 for each one (government regulated). Of this €0.30:
    a. €0.15 will be given to the customer.
    b. €0.05 will be kept by you. (For investment and maintenance of machines)
    c. €0.10 will be added to the total lottery pool.

3. Consumers can redeem their credit at a cash register.

To illustrate a full example, imagine 1000 people each donate 20 bottles to the same machine over the course of a full week. The lottery pool will then consist of €0.10 * 20,000 = €2,000. Since each consumer donated the same amount of bottles, they each have an equally likely chance of winning the lottery. One of the 1000 people will then be chosen to receive the €2,000 euros and it will be credited to their account.

Technology (CW)

To build a system that parallels the recycling machine, we used an Android smartphone. The smartphone provides a fingerprint scanner for identification and a camera for object recognition, namely waste recognition. To recycle a bottle, the user either logs in or registers an address on the Ethereum blockchain using their fingerprint. They then deposit their waste, which is promptly identified using computer vision and artificial intelligence, and collected. Their wallet is then updated on the Ethereum blockchain to reflect the ANSO coins earned. After a fixed number of bottles have been recycled, a smart contract generates a random winner based on the number of ANSO coins in their account, and the token amounts for all addresses are set back to zero. To collect the money, the user can scan their finger at a collection point (in most cases, this would be at the cash register of a grocery store, as it is done now).

Blockchain

Smart Contract

During the Hackathon, we decided to deploy and test our smart contract on the Ropsten Ethereum Test Network. It's worth pointing out that the fundamental idea can be implemented on any other blockchain such as NEO, QTUM or similar blockchains which support trusted decentralized applications (dApps).

In the Ethereum space, smart contracts are written in Solidity, a contract-oriented, high-level programming language specifically made for Ethereum. A contract in the sense of Solidity is a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain.

Our smart contract simply provides a function, called `recycle()`, which is invoked by the recycling machine to increase the amount of recycled bottles of a specific user. The function has a few requirements in order to enter the actual logic (e.g. is the lottery funded, does the sender's address not equal `0x0`). If the requirements are satisfied, we increment the corresponding user's recycling balance as well as the lottery counter. In case the latter reaches the lottery

threshold, we pick a winner with a weighted probability depending on the number of bottles recycled.

There are also a few other functions that are not related to the recycling process and therefore not worthwhile to discuss here.

Oraclize (PI)

Generating random numbers on the Ethereum blockchain is more complicated than in other programming environments (where you might use Math.random() or another related function). Our solution was to use Oraclize's built-in random number generator to generate the random number used to choose the winner of the lottery each week.

Due to the asynchronous call to Oraclize we had to introduce two states, namely `Recycling` and `PickingWinner`. Their purpose is to keep track of the current state of the internal state of the contract. Once the jackpot is hit, the contract changes the state to `PickingWinner` and calls Oraclize. As soon as the contract received an answer and a winner is picked, the state is changed back to `Recycling`.

We have also built in a fallback in the case the Oraclize call times out or the proof verification of the response fails. In such case, the winning number equals the hash of the last block number modulus the jackpot.

Security (PI)

Smart contracts operate on sensitive publicly visible data and transactions. This makes it crucial to implement resistance against malicious activity when developing contracts. To achieve this we complied with the security recommendations described in the official Solidity documentation.

One of the most common security pitfalls is that of **re-entrancy**, which means that an external contract can do callbacks to the callee contract before interaction has completed. This can potentially lead to serious problems such as multiple fund withdrawals, integer overflows, etc. To prevent this, we use the **Checks-Effects-Interactions** pattern. It states that we first perform all checks on the function to be executed (checking caller, available funds, etc.). If all checks pass, we apply the effects on state variables of the current contract (updating values, adding/removing entries, etc.). As a final step all interactions with other contracts are executed.

As explained above, generating truly random numbers in Solidity is not a trivial process. For this reason we contact an external oracle through the use of Oraclize services. This consists of two parts. First, the source of the random number is a trusted Ledger application integrated with Ethereum. Secondly, we ensure that data in transit (the random number) has not been tampered with before being delivered to the smart contract. This works by enforcing an onchain authenticity proof to be conducted, so that data which is not backed up by a valid proof does not get executed in the callback. The authenticity of data can be verified by the EVM.

Mobile Application

Design (JPS, AU)

Simplicity and an easy-to-grasp workflow were the main design goals in our application. Users should quickly accomplish the most important user interactions, i.e. getting an overview about their current deposit & lottery balance and scanning recycling items. This implies reducing the amount of choices asked during each step.

When the application is opened, there are two options available: register a user, and log in (for an existing one). Initially, each user should be registered in our system using the fingerprint. In the registration phase, the application creates a new account, that is uniquely identified by the fingerprint. We used this biometric authentication method as we want to make the process of recycling as quick as possible for the user, increasing usability and security in the same time.

After the fingerprint authentication, the application directly shows the current account balance, announcing a possible lottery win. The user immediately recognizes the main interaction point offered by the prominent “scan” button. A full-screen camera live picture allows the user to concentrate on the task to recycle items.

Image Recognition (JPS, AU)

As we have shown in our demo during the presentation, to avoid cheating, we use image recognition to detect if a person is indeed recycling a plastic bottle. When a user presses the Scan button, the application takes a picture, analyzes it, and returns the decision message. Thus, with this feature we simulate a real bottle recognition system that will be placed on the real recycling machine.

In order to perform this task, a cloud-based image recognition based on the Microsoft Azure Computer Vision API is used. It offers an extensive RESTful API to analyze features in pictures, e.g. descriptive tags, text recognition or landmark identification. In our example, we only request tags. For this, the camera image output is sent using a `POST` request to the analytics servers. A JSON object is returned where the matching keywords are searched on the end-user device. In our test case, these are “cup”, “glass”, “water” and “bottle”, but can be expanded to the final use case scenario.

Our cloud-based approach has the advantage that a proven back-end away from the user’s control is used. The provider can react globally to any changes that might be necessary because of wrong or spoofed results. Ideally, a trusted server running image recognition algorithms is used to mitigate privacy concerns of relying on an external provider. To further increase the detection rate, the final system should use an image capturing environment with known light conditions and additional sensors. Especially an infrared- or multi-camera-based

depth recognition system will lower false-positives as the shape of the recycled items can be detected.

Fingerprint (CW, JPS)

Registering and authenticating users on the blockchain can be accomplished through a number of methods. We decided for users to access their wallet using their fingerprint, instead of requiring users to have a smartphone/other hardware. This allows for users in situations of extreme poverty to use our system.

From a technical side regarding our project, the fingerprint authentication is done using Android's inbuilt `FingerprintManager`. It offers a function call to `authenticate()` which takes a `CryptoObject` as input, i.e. an cryptographically signed object reflecting the authentication state. This object needs to be signed by a `Cipher` guaranteeing its immutability. The fingerprint signature itself is taken from the inbuilt Android key storage.

A limitation to our use-case scenario is the lack of multi-user identification: we can only check whether the fingerprint belongs to the current user. Thus, the final system should add a central signature storage. For privacy concerns, this authentication storage should only contain hashed data to make a reconstruction of the underlying fingerprint infeasible.

Connection to Blockchain (JPS, AI, AU)

As an underlying library, `web3j` has been used to interact with the deployed smart contract. Although it was designed for the interaction with the Ethereum blockchain on Java and Android, in our case it did not work on the Android app due to some incompatibilities, therefore we had to use the Java version. We added wrappers allowing us to easily address the main user functionalities implemented in our smart contract, i.e. getting the current balance `getBalance()`, checking for a lottery win `didIWin()` and of course recycling new items `recycle()`. Additionally, administration tasks like setting the jackpot amount or information about the lottery funding have been implemented, allowing to extent the mobile app to more use-case scenarios right away.

All these wrapper functions, called from the Android application, communicate securely with an external server that is deployed on a virtual machine in Azure. Because we had to use Java for the web3j library, we have decided to run a socket server to which the Android application will communicate. We have designed a protocol for effectively and securely transmitting the intentions from the Android application to the socket server. With extensibility in mind, we analysed the current cloud platforms that are widely available, in order to pick one that will satisfy the platform needs in the future. The next table contains our final decision, the alternatives and the arguments for our decision.

Decision	The server will run on Microsoft Azure.
Alternatives	Google Cloud Google Cloud offers very competitive services compared with Azure with similar prices, but does not offer as many instance types as Azure which leads to a disadvantage in terms of scalability Digital Ocean Digital Ocean is best suited for small projects, as it does not have such a large offer of cloud services.
Arguments	Azure provides the widest ranges of services between the three nominated choices, also it has a widespread geographical server distribution that will assure the performance requirements needed by our critical system. The system running on servers located closer to the monitored area and reduce latency, therefore Azure is the chosen solution.

Summarizing the workflow of the Android application, when a user logs in into his or her account using the fingerprint, the methods *getBalance()* and *didIWin()* are invoked. These methods send a request to our external server, that is deployed in Azure, and runs web3j library. Once the server has the balance value and the lottery status, it will send these values back to the user's mobile application. When the user scans a new object, the picture is sent to the Azure Computer Vision API that will reply with the name of the objects that are in the picture. In case of success (there is indeed a plastic bottle), the application calls *recycle()* method which will notify the external server to increase the balance of the current user. The application does not wait for the operation to complete (to increase the user's balance on etherium) because this operation is time consuming. Once the update request has reached the server, the user can close the application and the server is responsible to do the update. Even if there is a faulty network, the server will try to perform again the update until it succeeds. This is the reason why a user gets the notification for winning the lottery at the next log in.

Furthermore, to exemplify how web3j works, I am going to describe the process of managing smart contract through a Java interface. Now, in order to interact with the Ethereum network, you need to have a local Ethereum client running, in order to be up to date with all the information in the network. Luckily, there are services that allow you to go past this inconvenience, such as Infura, which provides free clients in the cloud. All you need to do is to start a free client on the cloud, by using the following command:

```
Web3j web3 = Web3j.build(new InfuraHttpService("https://ropsten.infura.io/yjAnYyZgUECNLVLUPpGK"));
```

Now, that we have a working client, we can start sending request to the Ethereum blockchain. In order to integrate with our smart contract, we first need to compile our Solidity written smart contracts into our Java programme. In order to achieve this, we are using the "solc" command line tool to compile our Solidity smart contract, then, we are using the web3j functionality to generate a wrapper around the contract, that can be used for further interactions within the network. Next, a contract can be either deployed or loaded from the network. All this is

integrated into the Socket server that interacts with the mobile application. More about the scalability of this in the next section.

Future Work (JPS, AI)

Our proof-of-concept implementation already allows to showcase the general workflow of our idea. Transforming it in a real-life application, we imagine extending current recycling systems to Internet of Things (IoT) platforms. Keeping the current image recognition hardware, the systems will be able to connect to a central authentication and analysis server using mobile networks. Thus, previous offline functionalities will be assisted by more advanced tasks possible due to the powerful backend. In turn, the systems will profit from the reliability of the proven recycling machines, but gain the flexibility thanks to the internet connection. To mitigate the impact of failures in the latter system, caches will store any changes that have not yet been synchronised with the backend. The approach of extending current systems not only minimises the initial costs, but also allows the save operation of the most critical components.

Moving to the scalability of our system, we have identified a major bottleneck that will make us unable to scale to many customers. The bottleneck the connection to the blockchain, more specifically the way we handled the communication between our Android application and the server that forwards the requests to the blockchain. Because the Android implementation of the web3j library is not ar reliable as the Java version, we will probably stick with a server side implementation of the connections, but instead of using simple Java Sockets to communicate with all the app users, we are going to use a REST API. By simply using HTTP requests, we can make the number of using interacting with the server virtually limitless, by either scaling up or horizontally.

ANSO Features (CW)

ANSO can be considered a Freigeld coin. It is spending-power stable in the sense that there will be no deflation/inflation associated with the currency. The currency has a fixed supply and the tokens expire after each lottery drawing, thus deflation/inflation would, in theory, not be an issue. ANSO is also cash-flow safe as the supply is constantly zeroed and redistributed in accordance with the lotteries. ANSO is able to be converted into other currencies by means of the lottery and any other markets created by demand to pay fiat currency for a higher chance to win the lottery. It is also localized to the area the machines are located, therefore it fulfills Gesell’s requirements to be classified as Freigeld.

The motivation behind ANSO being Freigeld is to ensure there is no incentive for users to amass the tokens without spending them, thus causing a supply choke. Since the tokens are redistributed every week with every lottery, the tokens usage is forced.

Psychology behind ANSO (CW, ASM)

Gambling is a potentially destructive habit that takes advantage of the brain's reward system to generate profit for an opportunistic actor. We take advantage of this human behavior to generate a positive outcome for society as a whole.

Looking to lotteries like Swiss Lotto for example, the jackpot is worth 3.8Mio CHF and the chances of being the lucky winner is 0.0000000317715337. If we calculate the expected value of playing the lottery, it is clear that it would be far more rational to save the money that would have been spent on the lottery ticket (5 CHF), but the prospect of having a nonzero probability of winning a large amount of money is far more enticing than saving small, predictable amounts of money. In the case of bottle recycling, recycling even one or two bottles may not be considered worth it in the traditional model, as it is "only" €0.30 or so, but with our system the chance to win a significantly larger amount of money should increase demand for recycling, as it gamifies depositing waste and follows an already-proven lottery model.

Since gambling gamifies the recycling process, and it has been proven to be effective in a lottery setting, our model is highly likely to increase the motivation of the higher-income population who might otherwise not be incentivized by the prospect of receiving a few cents for each bottle.

Conclusion

Extensibility (CW)

Currently we focus on the recycling of PET. It is then relatively straightforward to upgrade existing recycling systems with our solution. In addition to PET, it is possible to extend to other waste sources: metal, glass, electronics, etc. For each category of waste there could be a corresponding token, and there could be an exchange that allows users to trade tokens generated from different forms of waste to maximize their chances of winning one specific lottery. If the recycling machines were equipped with my complex sensors, then a number of materials could be verified and recycled (even materials that are not normally compensated for in the form of a reward). For example, a recycling machine could be set up to receive batteries or mobile phones and dispense rewards accordingly.

One method of including the producers of the plastic waste into the system is to introduce a second token. The producers could be given a fixed amount of tokens which correspond to the amount of waste they can produce. In order to produce more waste, these producers must buy back more tokens from the consumers to produce a closed circle on the amount of waste that is

allowed to be generated. Modifications to allow for growing population, demand, etc. can be added by imposing additional costs on the increased amount of bottles that would need to be produced. In an ideal scenario, the costs borne by the producers would correspond to an increase towards spending on recycling sufficient enough to offset the negative externalities of the additional bottle being produced.

One clear way to enforce cost impositions on the producer would be through regulation. If government intervention is a possibility, it can be a requirement that the producers of the waste must pay for tokens as a tax, and this tax can be used to fund the recycling rewards, This would create a method for a closed-loop cycle of waste and bottle production, incentivizing recycling and disincentivizing overproduction.

Another way to view our solution is a system that can place a “bounty” on a certain material, which in turn drives people to deposit this material into our system in exchange for a potential large monetary gain. Even non-waste material could be collected using our system. Precious metals like gold and silver, commodities like coffee beans, natural resources like water, could all be collected. A dynamic reward model that varies the amount of tokens with the demand for that material could be incorporated to moderate the users’ behavior, and to drive collection for specific materials. This would allow for entirely new methods of sourcing and could realistically open new areas of opportunity within the private sector. It may sound like a bit of a stretch, but the possibilities could be reasonably envisioned.

Efficacy Testing (CW)

In order to implement the system (to then enable efficacy testing), we need to partner with local businesses, ideally located in Germany, the US, or Sweden, who offer rewards for recycling at a system as we detailed previously in the report. A few of these machines could be updated themselves by changing the machine’s internal code, however this would prove to be more “hacking” than necessary. A more straightforward approach would be to add a mechanism for consumers to collect their tokens by entering an address they receive printed-out on their receipts at the cash register into a website that allows them to claim the tokens to a specific address. This would require less intervention from a technical perspective, and would also encourage active participation into the lottery. We have seen this system used in the past with certain soft drink companies, where they print a code on the cap of each bottle that can be entered online for a chance to win various prizes. This proof-of-concept implementation would then ideally be upgraded to an automatic token-reward system once the idea has been validated through the initial tests.

To be able to test the efficacy of our implemented system, we could look at the year-to-year change of plastic bottles being recycled. This can be compared against plastic bottle consumption rates in order to obtain the percentage of bottles consumed that year that have been recycled.

It might be that the incentive could work too well, in which case we would actually incentive people to increase their consumption of plastic bottles. These bottles would, however, be returned and thus be recycled, so the problem is not as severe. Ideally, plastic bottle production would slowly be phased out due to its deleterious effects on the environment.

If our system leads to significantly increased consumption, producers could be asked to pay into the token incentivization system, as a tax on their increased sales volume. This could then be used to offset the negative externalities of the increased bottle production.

Outlook (CW)

All in all, our system provides a fun way to incentivize recycling using blockchain technology, artificial intelligence, and what we've learned from behavioral psychology as it pertains to reinforcement. It is easy to implement (especially upon existing recycling infrastructure) and the possible impact is substantial, making our solution both high-tech and high-potential.

Sources

1. Plastic recycling facts and figures. *The Balance* Available at: https://www.thebalance.com/plastic-recycling-facts-and-figures-2877886. (Accessed: 14th February 2018)
2. Parker, L. A Whopping 91% of Plastic Isn't Recycled. *National Geographic* (JULY 19, 2017). Available at: https://news.nationalgeographic.com/2017/07/plastic-produced-recycling-waste-ocean-trash-debris-environment/. (Accessed: 14th February 2018)
3. Fabian Meyer, J. B. The Closed-Loop Endeavour - A Case Study on Barriers and Enhancements of the PET Bottle-to-Bottle Recycling Systems in Germany and Sweden. (2016).
4. Container deposit legislation. *Wikipedia* (2018). Available at: https://en.wikipedia.org/wiki/Container_deposit_legislation. (Accessed: 14th February 2018)
5. Thörnelöf, I. Increasing Recycling through Container Deposit. (Uppsala University, 2016).
6. Fründt, S. Das Milliardengeschäft mit dem Einwegpfand. *Die Welt* (2007). Available at: https://www.welt.de/wams_print/article883460/Das-Milliardengeschaeft-mit-dem-Einwegpfand.html. (Accessed: 14th August 2017)