ETH Zürich, 30.04.2018 | BIOTS 2018

UBS: Trusted Tracking and Alerting

Team report

ANDRICOPOULOS, Ari Dimitri - aari@student.ethz.ch
BIRRER, Vera - birrerve@student.ethz.ch
BRELY-DEMEULES, Justin - justinb@student.ethz.ch
LÜTHI, Noah - noluethi@student.ethz.ch
MILEWSKI, Marc - marcmi@student.ethz.ch
WACKER, Pascal Alain - pwacker@student.ethz.ch

All contributors contributed equally to this report

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

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.

Table of Contents

1. Introduction

The problem

Imagine you are a seller or buyer of international goods. These could be sensitive to cold or hot temperatures or maybe vibration, and they could be of high value. You or your clients may receive damaged goods due to rough transportation. Sometimes you may find torn open or broken packaging and missing pieces. Due to the expensive nature of the goods, sometimes items may have been replaced in transit with similar looking ones but of inferior value.

UBS are seeking a solution allowing goods in transit can be tracked, protected against fraudulent activities and their status monitored for compliance with the required transportation conditions.

UBS wishes to be informed and warned clearly, continuously and in real-time of any deviation from the target state. UBS is open to the use of blockchain technology to help ensure standards are adhered to. How can an (existing) IoT device/ sensor be linked to the blockchain to keep track of goods and provide above status updates?

The idea

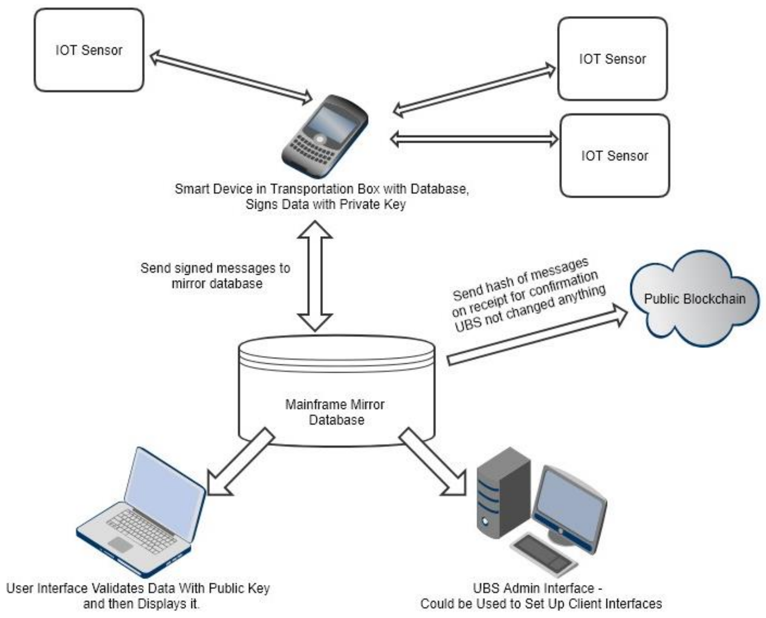
The idea is to create a flexible system for monitoring transit conditions, with a high level of security regarding data integrity. We call this UBS Black Box.

The set-up

The proposed set-up is as follows.

A system of sensors is connected to a monitoring device. The monitoring device is linked to a UBS controlled mainframe database. From this we will have a user interface and admin interface. The data will be secured with public-private key cryptography linked in a chain as well as blockchain technology.

The idea is that the system is agnostic to the type of sensor used. There are two types of messages from the sensors – monitoring messages (eg. temperature given at set intervals) and alert messages (eg. "This box has been opened"). All messages are stored on the device and signed with a private key in a way that links in a chain with the previous message, thus ensuring both a) that all messages are correct readings and b) no messages are missing. The messages are also transmitted whenever possible to the mainframe using a database mirroring protocol. The user can thus have real-time monitoring of the condition of the goods.


IOT Sensor
IOT Sensor
IOT Sensor
Smart Device in Transportation Box with Database, Signs Data with Private Key
Send signed messages to mirror database
Send hash of messages on receipt for confirmation UBS not changed anything
Public Blockchain
Mainframe Mirror Database
User Interface Validates Data With Public Key and then Displays it.
UBS Admin Interface - Could be Used to Set Up Client Interfaces

In order to ensure that no-one at UBS tampers with the data later, a hash of every new message sent to the mainframe is also stored on the blockchain – giving it an unalterable timestamp.

At the end, the user officially 'ends' the journey on the user interface and all of the data is verified using the public key and displayed in a useful format to the client on the user interface. It could form the basis of a 'smart contract' on the block chain.

In diagrammatical form this looks like the schema above. Moving from top to bottom,

1) The IOT sensors transmit Bluetooth messages to the smart device which is located in the transportation box.
2) The smart device signs these messages with its private key in a 'chain', whereby the previous signature is used as part of the formation of the next hash. This ensures a cryptographically secure order of events.
3) These messages are sent (with the signature) to a mainframe mirror database. This takes place whenever the smart device is in range of mobile internet.
4) The mainframe then sends hashes of these messages to a public blockchain, effectively timestamping the receipt.
5) Both the user and UBS admin can then look at the information, safe in the knowledge that the messages are very unlikely to have been tampered with – either on the mobile device or the UBS mainframe.

In the rest of the report, we will go through the parts of this system. We will seek to identify possibilities and then suggest what we believe to be a good way of implementing the ideas above.

2. IoT Sensors – possibilities

There are a number of possible sensors that can be used. Below a few potentially useful ones are listed.

Position tracking sensor

Position tracking is a basic requirement of this system. All versions will use this. We want to be able to locate the package as reliably as possible in order to keep the client aware of the location.

There are a number of different methods for tracking position, but the GPS standard has emerged as the leading protocol here. GPS tags establish position by analyzing the distance of the tag to several GPS satellites in orbits (with known location).

It is important to note that the energy consumption of GPS is higher than that of RFID tags since GPS requires a power source which will supply the processor calculating its location. Nevertheless, RFID does not have the range of a GPS tracker and hence cannot replace it. In our scenario, deliveries can span from one point of the world to another, so GPS is more suitable. All smart phones use GPS, and we would just be able to use a device based upon current mobile phone technology. Therefore, a GPS tracking device would probably be built in to the main portable device.

Temperature sensor

Certain products require a careful tracking of the temperature of the environment in which it is exposed. Obvious examples of such goods are crops, foods in general, pharmaceuticals such as pills and medicine etc. Exposure to non-conform temperatures can cause a multitude of consequences, from rotten food to spoiled vaccines or antibiotics, which can be lethal. These sensors can be configured in a way such that the client (or the sender) sets a determined range in which the temperature can fluctuate. This way one can confidently know if the order is being handled in conformity. Moreover, by delivering the product in its intended state, the amount of waste is reduced.

Motion detection sensor

In this case, the device tracks the physical movement of the object to which the sensor is attached or any motion in its environment in general. With such an instrument, we can get an indication that someone is in proximity of the goods and with that information we can analyzing if it seems plausible that the motion detector has been triggered (delivery has arrived at the client, package is exposed to a customs control, etc.). Additionally, this piece of technology can be used to detect the movement of the products and determine if they were displaced or even damaged.

Pressure sensor

If the client's delivery must be transported by air and/or the goods are vulnerable to pressure (in form of high altitude or the weight of stacked products, etc.) the use of a pressure sensor can be practical. This accessory senses the pressure by using a force collector (diaphragm, piston, bourdon tube) to measure the strain due to an applied force over a surface area. Following the same concept as the temperature sensor, the client can define a range in which the pressure can be considered as acceptable.

Image sensor

With these sensors, you can store optical images in an electronic signal, which then can be sent like other information to the blockchain. There can be CCD or CMOS image sensors, which are different in technology, but they do not have an important quality difference, so it doesn’t really matter, which of them will be used. These sensors are located in small cameras, which could film in case 1 the inside of a box or also in case 2 from the inside to the outside of the box.
In case 1, the box can be supervised, if something unexpected happens, whereas in case 2 for example the opener of a box can be filmed / photographed and be recognised or at least be analysed through these image sensors. It is possible to capture the race, gender, age approximation.

Accelerometer

In case fragile goods susceptible to shock are transported it could be necessary to surveil the inertial forces acting on it. This could possibly be built into the main device, as the acceleration will probably be the same anywhere in the box.

Optical sensor

It might be interesting to surveil the amount of light if goods susceptible to light are transported or as a manner of intrusion detection. A change in light could mean that the box has been opened.

Smoke sensor

In order to trigger an alert in case of fire and register possible sources of pollution there might be a need for smoke sensors. There is a possibility to attach smoke sensors to the system.

Chemical/Gas sensor

To monitor pollution or possible leakages if chemicals are transported, there is the possibility to use gas sensors sensitive for specific agents.

Proximity sensor

This sensor essentially controls the presence or absence of a nearby object. In other words, it verifies if a distance is held or that a certain range is free of access (a garage entrance for example). To implement in our study case, this device can be used in a way to make sure box remains closed during transportation, and, if not, be noticed so the owner (UBS) can be informed. Additionally, the sensor can be responsible to keep track of the distance to another nearby box; this way we can be updated to any from of movement in space (eg. the box itself or nearby has been lifted).

An example of the types of sensors we might wish to use

EXAMPLE: www.ti.com/sensortag
In Appendix 1.1, is an example of a smart device, both in and out of protective case.

It costs around $20 when bought in large quantities. It communicates via Bluetooth to the main device. This sensor measures temperature, infra-red, light, humidity and pressure; just one sensor provides a lot of information. In terms of battery life, it will last about a week with data sent every 5 minutes or 6 months with one message per day.

It can be set up to monitor quite often (eg. every second) but send data every 20 minute or every time some event happens (like temperature goes from below 25 C to above 25 C). It is a programmable device so whatever is require can be requested. If we want fewer messages on lower battery this is programmable. Devices like this can work on either Wi-Fi, Bluetooth or SIM card. They could also connect with wires, but this is not usually preferable.

3. Communications

a. Communication between sensors and device

The communication between the sensors and the 'BlackBox' is critical for the project. Once again, a wide variety of alternatives are available from Wi-Fi, Bluetooth, SIM or NFC to simple cable wiring. For practicality reasons, Bluetooth seems to be the best choice. First of all, it is wireless, so you do not have to cable management or cable erosion into consideration. In the case where sensors are not located next to the 'BlackBox', it becomes paramount to have significant range. Other advantages are that Bluetooth is highly energy efficient compared to Wi-Fi. Furthermore, it does not demand additional infrastructure to operate, whereas with Wi-Fi one must implement a router to communicate. Obviously, the bandwidth is lower when using Bluetooth, however, assuming your sensors only transmit data when there is a change in information (compared to the previous measurement) ~1Mbps is sufficient.

To simplify and lighten data transmission even more, MQTT (Message Queuing Telemetry Transport) can come in handy. It is basically a very simple publish-subscribe protocol. By using it, we can ensure a longer battery life since the main reason for short lasting (yet practical) power sources is excessive data transfer.

b. Communication between device and mainframe:

We took a look at different technologies to get data from our 'BlackBox' to a server hosted by UBS. The device sends its sensor data both periodically as well as when it gets triggered by an external event (ex. box opened, temperature rise, and so on). The data is stored in the box as well as the server and gets confirmed at the end of the shipment.

Short range technologies like ZigBee or BLE don't work in our use case, as the don't have a large enough range. Additionally, depending on the technology, they can consume a lot of energy. We could overcome that limitation by deploying an array of bridge devices and form a mesh network, however that would neither be practical, nor would it make sense economically.
This leaves us with 3 technologies. LPWAN (Low Power Wide Area Network), VSAT (Very Small Aperture Terminal) and cellular networks (SIM Cards).

For LPWAN technologies we could either use LoRa which has a range of ~10-20km, has a broad support (in Switzerland Swisscom offers a nationwide network infrastructure). The competitor would be SigFox, which has a slightly larger range, but not as good support.
The downsides of using LPWAN technologies is, that there can be "blind spots" where no connection is given (for example on a cargo ship at sea). If the shipping vessel (ship, train, truck, airplane) has some kind of connection to the internet, we could put an antenna on it and thus extend the LPWAN network dynamically to where the box is.
The second technology relies on a satellite connection. There are VSAT receivers that can be mounted on a truck or a shipping container. They offer worldwide coverage and are hard to disturb.
As an alternative to LPWAN and VSAT, we could also leverage existing cellular networks and either send data by GSM/LTE or fall back to USSD messages, which are hard to block and work on close to no connection. An approach using SIM-Cards has the advantage, that we don't need to setup any infrastructure and it works while the cargo is in transit. However, there are blind spots (remote areas, sea, air).
The SIM is almost certainly the easiest solution to implement. For almost all practical purposes it may not be necessary to have constant coverage. As long as the data can be checked on arrival (which it can) and the integrity can be trusted (which it can) adding further expense may be overkill.
However, the brief has said that UBS require real-time updates. If maximum coverage were required it may be required to deploy more than one technology, to have a certain level of redundancy and make it harder for malicious actors to block signals from going out.

Although we don't think constant monitoring is necessary, our suggestion for constant monitoring be to use VSAT in addition to the SIM solution, if the container is large enough to support it as there's a worldwide coverage. If the container isn't large enough, we'd suggest deploying both the SIM-card solution as well as a LoRa (LPWAN) solution. We'd suggest fitting the shipping vessel with a LoRa antenna and use its onboard communication methods (for example SIM-Card for a Truck, Satellite communication for a ship and so on) to get the data to our data center.

There may be issues because the device is stored inside a packing crate which may be stored packed in the middle of other packing crates. The signal may not be that strong with all of the shielding, and what should be done to overcome this depends on how urgently UBS needs updates.

4. Message encryption chain

This is an example view of what the database might look like:

Device	Journey number	Transaction Number	Sensor	Sensor Type	Date-Time	Location	Reading 1	Reading 2	Reading 3	Signature
123ABC	1002	1	3838rt	Temperature	2/14/2018 12:30	1.00988998,36.787849	13			7397839hjehk
123ABC	1002	2	3838rt	Temperature	2/14/2018 12:40	1.00988998,36.787850	14			28973jhkjhfk
123ABC	1002	3	3838rt	Temperature	2/14/2018 12:50	1.00988998,36.787851	13			182937hjkhk
123ABC	1002	4	7638ob	OpenBox	2/14/2018 13:00	1.00988998,36.787852	OPEN			123123hjkhjk
123ABC	1002	5	3838rt	Temperature	2/14/2018 13:10	1.00988998,36.787853	14			123879hkjhk

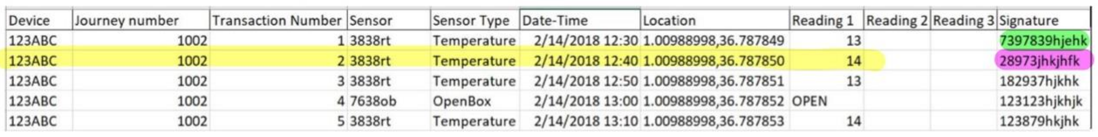
The moment the message is put into the database, it is signed by the device. Here each signature signs a message using a private key which is stored on the device. When new information comes in from a sensor, it puts a buffer in while the database inserts and signs the new information. In this way we prevent the chain branching into 2. The message to be signed consists of everything in the database line appended with the previous signature.

This is how public-private key signing would work:

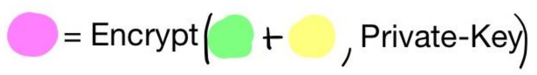
Encrypt(message, private-key) => Signature
Decrypt(Signature, message, public-key) => T/F
where
message = concatenate(Signature{L-1}, Info{L})

You are signing the signature from the line before plus the information in that line. You encrypt the message with the private key held only on the portable device. You can then decrypt on the user interface using the paired public key to confirm that data are correct.

A representation based on the dummy data is shown below:



Device	Journey number	Transaction Number	Sensor	Sensor Type	Date-Time	Location	Reading 1	Reading 2	Reading 3	Signature
123ABC	1002	1	3838rt	Temperature	2/14/2018 12:30	1.00988998,36.787849	13			7397839hjehk
123ABC	1002	2	3838rt	Temperature	2/14/2018 12:40	1.00988998,36.787850	14			28973jhkjhfk
123ABC	1002	3	3838rt	Temperature	2/14/2018 12:50	1.00988998,36.787851	13			182937hjkhk
123ABC	1002	4	7638ob	OpenBox	2/14/2018 13:00	1.00988998,36.787852	OPEN			123123hjkhjk
123ABC	1002	5	3838rt	Temperature	2/14/2018 13:10	1.00988998,36.787853	14			123879hkjhk



This method ensures that the information for each line is exactly as recorded live on the device. And because the previous line's signature is used in the next line's signature, it means that no lines can be deleted, or the decryption will come up as false. This means that the data is both correct and complete.
In this way by going through the database, line by line we can (using the public key) confirm that a) each line is as originally recorded and b) all lines are present; there is no missing data.

Buffering

In order to ensure the integrity of the chain, the device can only take one message at a time. A buffering will have to be put in place on the receipt of a message to ensure that any new message is added to the chain strictly after the last one. If this is not in place, then the chain can fork.

5. Mirroring and sending messages to public blockchain

Mirroring

We would use a mirrored database, such that all additions to the device database are mirrored on the mainframe. The mainframe database should be set up so that any changes are logged. If any data in existing table is changed, then this will give an exception in the main frame as this should never happen.

Timestamping in blockchain

When a new database line is put in it also gets hashed in the mirror database and this hash is placed onto the block chain. It is done in the following manner:

Set up an Ethereum contract with an array with two fields –
1) Date-time NOW (from the miner's system)
2) Hash of the line from database.
Every time a new line is added a function is called which adds a line to this array.

Now, if there is ever any doubt about the integrity of the UBS mainframe data, the entries can be externally timestamped. This means that they can't be later changed without it being obvious.

6. Hardware issues – where they go (brackets), battery life, charging etc.

Device hardware: A **Raspberry Pi** would be a good choice for hardware for the transmitting device. It has a lot of computing power (enough for our purposes) in a small volume of space. This is used by Ambrosius, for example, for their equivalent device.

Bluetooth uses less energy than Wi-Fi because of the communication protocol. It is slightly less secure for the same reason.

Some sensors need a lot of monitoring and thus use a lot of battery life. Vibration, for example, requires data that is very finely grained. These sensors need batteries changed more often.

7. Criminal exchange of goods – possible solutions

The problem of goods being replaced by inferior looking equivalents could be combated in various ways.

There exist stickers that cannot be removed without damaging the sticker (like a Vignette). One could put a very distinct difficult to copy sticker (eg. holographic) of this type on each good.

One could also set up cameras that only turn on when there is a journey in progress and there is light in the box. They could send this footage to the mainframe and it could be watched if there is any doubt.

Image sensors, which take photographs eg. every 10 seconds for 1 minute if light changes to above a certain level, can be used.

8. User experience – start to end of trip.

Box set-up

Each packing box needs to be set up with whichever sensors are required. These need to be linked to the monitoring device using software installed on the device (which is compatible with certain predefined IoT devices). You can set up a bracket for holding the monitoring device within the box.

Sending package

With IoT sensors in place, and the device safely in place, your box is ready to be filled as normal. When the box is about to be closed, set up a new journey and press the "Start"-button. Can also be started remotely by the sender.

Deterrence of theft

Stickers can be placed on the box, showing that the cargo is protected by UBS 'BlackBox'. This should act as a deterrent to would-be thieves, in the way that a burglar alarm can act as a deterrent even if it is not on. Also, this would alert security people that the electronic equipment inside is part of the security system and not something more sinister!

Package receiving and turning off the device

When the package is received, the recipient can go into their app and end the journey. For security our device can't be turned off by just pressing a button on the app as someone with something to hide might do that and blame the device for turning off.

If data were to look incomplete on the mainframe database, you could also plug travelling device into your computer to check the data from there. This should be unnecessary.
If the box has been opened, then if the goods are individually fitted with special hologram stickers these can be checked. Items counted, etc.

Note that the sender can 'start' the journey remotely; the receiver can 'end' the journey remotely. This would be in case they were not there in person or forgot to start it at the time.

Warnings

The user interface can give alerts if some prespecified limits have been breached. If there is a warning from a sensor, the goods need to be inspected. It could be an indication that the goods can't be sold any more (particularly for perishables and pharmaceuticals).

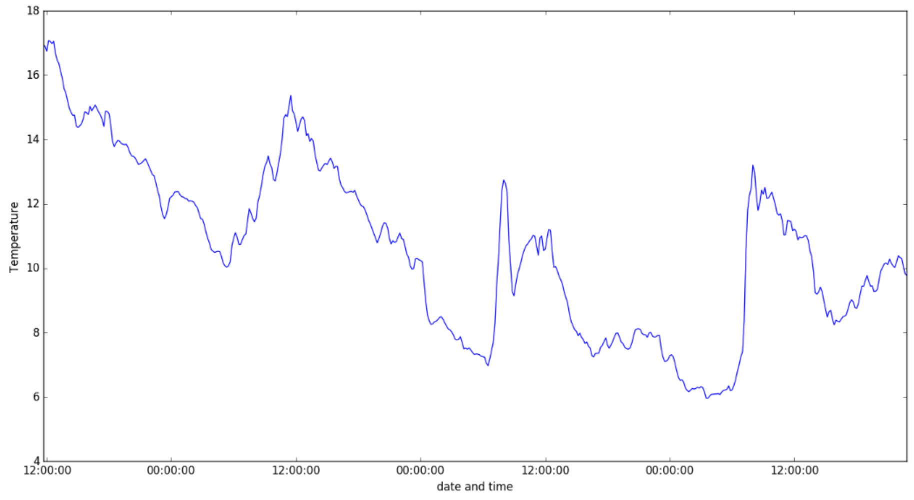
Diagnosing problem points

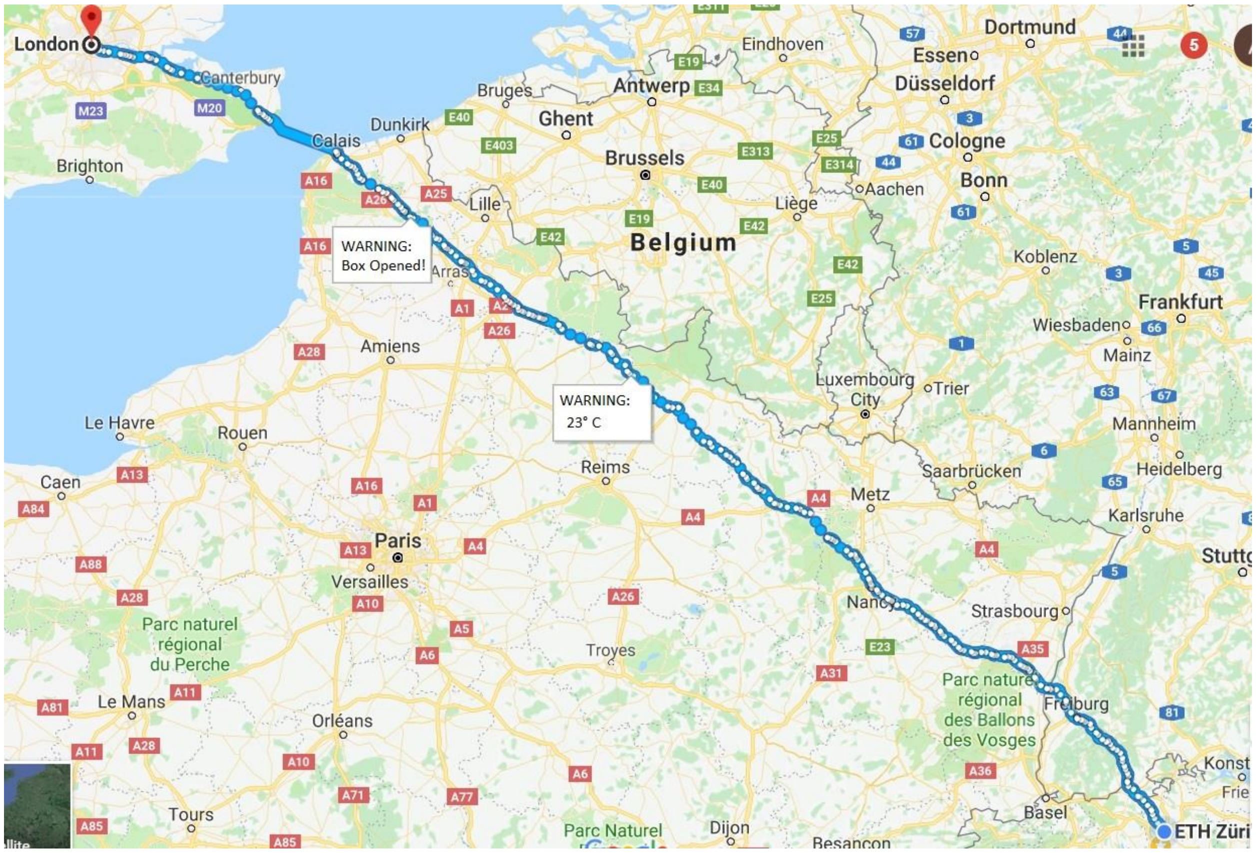
From looking at the location data and the sensor data it should be possible to pinpoint exactly where any damage happened.

The sender has access to this data to help them with their logistics.

9. User interface

The information from the black box is shown on the user interface. There you can look up the location over time, and all the readings from the sensors. You can see graphs and maps. You can be sure of the integrity of this data thanks to the public/private encryption.





10. Extensions: cameras, border control, smart contracts, small scale test

Cameras

No sensor system will be fully secure. Also, even if you can detect a theft you still won't know who perpetrated it. Cameras can also help with security, acting both as a deterrent and a help to find those responsible. Use of image sensors that automatically take a photo when something changes could be a cheap way of implementing this.

Smart contracts

To go further in the program in trust tracking you can go into the world of smart contracts on the blockchain, for example the Ethereum blockchain.
There will be contracts between the sender and the receiver. The sender can check if the receiver already has paid the money into the contract before delivering the goods. The receiver only pays for the goods if they arrive safely at the destination and all measurements are within the ranges agreed in the contract.
The whole transaction could be with cryptocurrency or some crypto-version of cash, and all is transacted totally automatically via the contract. UBS could help their clients by setting up the contracts and charging appropriately.

Border control

Border control is a problem for smart contracts. If customs insist on opening something on the way, then we need a way to override a smart contract setting that says it can't be opened. At the same time, this should not be open to abuse – people who open the box at other times should not be allowed to call it a customs opening. Location can help here. Customs could also have a code sent to them using some kind of two-factor authentication like internet banking. They could type this code into the box to indemnify the opening against smart contract failure. This is for further research.

How to get proof of concept:

A small-scale test of this would involve an app on eg. an Android phone, which connects to various IoT sensors and stores data in a database, mirroring the database. The main questions would concern reliability of the sensors and connections.

If this is successful, and a good view of the journey is achieved, the rest should be just a matter of software development. The encryption and blockchain part is pretty straightforward, and the user interface just displays the data in a certain format.

11. Risks – from the start to the end of a delivery

Hacking of devices, intercepting signal:

This is always a risk, but the work to do this is very large. Ways around it involves installing more sensors as the work soon becomes prohibitive and inconsistencies will appear.

Messing with sensors

One could physically change conditions close to sensors eg putting ice cubes on devices. This can also be deterred by adding more devices. But non-sensor methods can also be used - eg cameras, security guards. In the end the solution, if important, will probably be a combination of cheap sensors and more expensive security. But the sensors save a lot on expensive security.

Batteries expiring

Since battery life of sensors and devices is largely dependent upon how many messages are sent, we can slow down monitoring if we have less battery life remaining. Monitoring battery life on devices and sensors is important -we should be able to monitor battery life at eg. 6 hourly intervals. Low batteries should be replaced/recharged before big journeys.

12. Code

We have created some demonstration software; the contract is written in Solidity, the Ethereum programming language; the rest in Javascript. The details are here:

GitHub Repo: https://github.com/ETHBiots2018/blackboxes

Demo Page: https://pascalwacker.github.io/blackboxes/signin.html (use any email/password you like, make sure to have MetaMask installed and unlocked before trying ;))

We have put some screenshots in Appendix 1.2.

13. Conclusion

Right now, freight traffic can often be opened and tampered with. Conditions inside the package may not be well monitored, and often this is important – for example with pharmaceuticals or perishable food. In these case, neither the sender nor the receiver has an idea where a package has been opened, in what condition the package is or if something unusual has happened on its journey. Companies often provide a tracking system, so you can know the location of your package. With our solution you have the ability to either check different measurements from built-in sensors, or under special circumstances supervise your product from the image sensors.

With different sensors in the box, you can know if something unusual is happening. Mostly this will be in real time, but even when it isn't block chain technology is used to guarantee the integrity of the data from the sensors. The security around this means that the user can totally trust the data as the real readings on the sensors.

In fact, the data integrity is so high that a smart contract could be enacted based upon it, whereby payment is held in escrow and automatically released upon delivery based on certain conditions being fulfilled.

The ensuring of good standards can only be good news for everyone. The deterrence of crime and carelessness provided by the blockchain powered solution, is a great help to buyers, sellers and for UBS as trade financer.

14. References

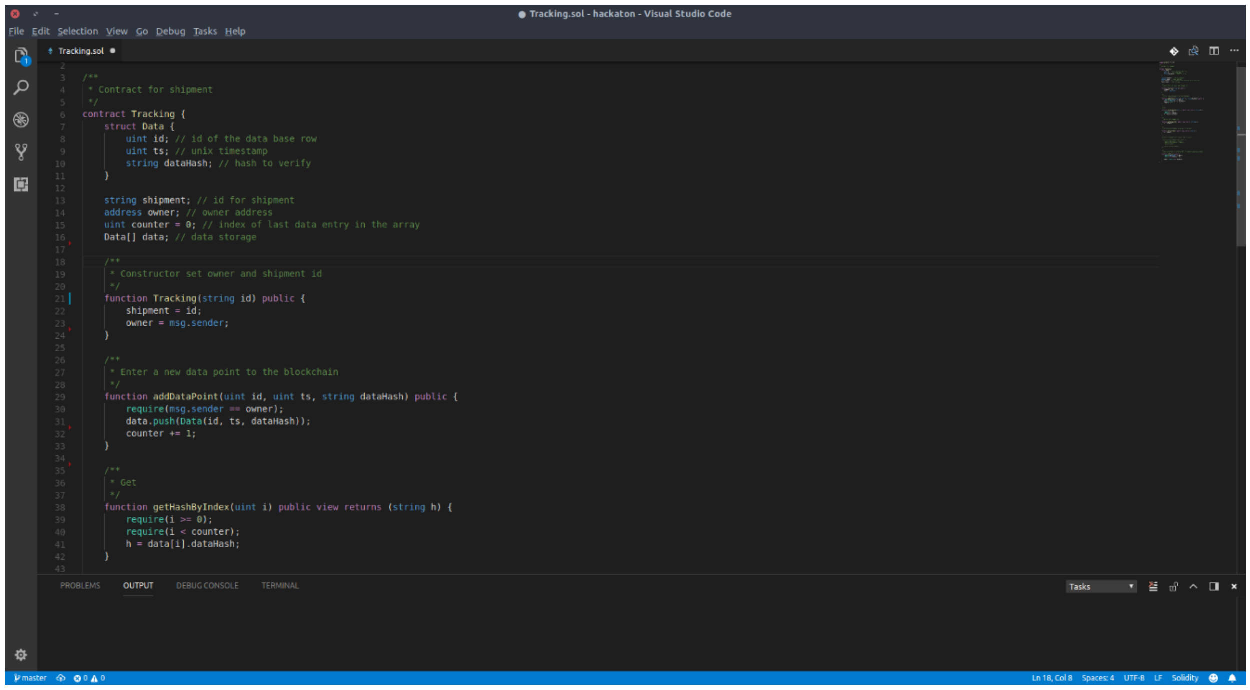
- Bluetooth: https://www.diffen.com/difference/Bluetooth_vs_Wifi
- Bluetooth vs. Wi-Fi: https://www.diffen.com/difference/Bluetooth_vs_Wifi
- GPS vs. RFID: https://www.airfinder.com/blog/rtls-use-cases/gps-vs-rfid-comparison-of-asset-location-technology
- Introducing the MQTT Security Fundamentals: https://www.hivemq.com/blog/introducing-the-mqtt-security-fundamentals
- LoRa alliance: https://www.lora-alliance.org/
- LPWAN: http://internetofthingsagenda.techtarget.com/definition/LPWAN-low-power-wide-area-network
- MQTT: https://learn.adafruit.com/mqtt-adafruit-io-and-you/why-mqtt
- Sensor list: https://www.finoit.com/blog/top-15-sensor-types-used-iot/
- SigFox & LoRa: https://www.link-labs.com/blog/sigfox-vs-lora
- XDK Guide MQTT: https://xdk.bosch-connectivity.com/documents/37728/286250/XDK_Guide_MQTT.pdf/46a3f0a6-c6a2-4825-bf87-903231caba80

15. Appendix:

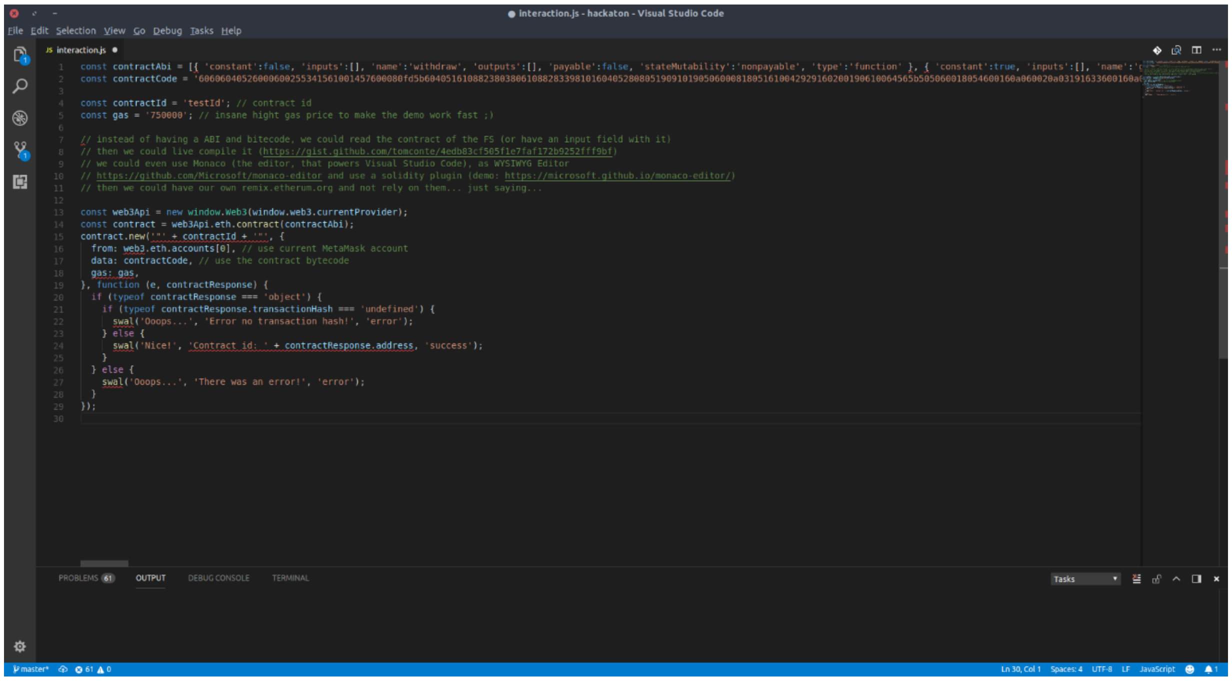
Appendix 1 – illustrations not shown in main text

1.1 An example of a smart device, both in and out of protective case.

1.2 Screenshots from code:



```
/**
 * Contract for shipment
 */
contract Tracking {
    struct Data {
        uint id; // id of the data base row
        uint ts; // unix timestamp
        string dataHash; // hash to verify
    }

    string shipment; // id for shipment
    address owner; // owner address
    uint counter = 0; // index of last data entry in the array
    Data[] data; // data storage

    /**
     * Constructor set owner and shipment id
     */
    function Tracking(string id) public {
        shipment = id;
        owner = msg.sender;
    }

    /**
     * Enter a new data point to the blockchain
     */
    function addDataPoint(uint id, uint ts, string dataHash) public {
        require(msg.sender == owner);
        data.push(Data(id, ts, dataHash));
        counter += 1;
    }

    /**
     * Get
     */
    function getHashByIndex(uint i) public view returns (string h) {
        require(i >= 0);
        require(i < counter);
        h = data[i].dataHash;
    }

```



```
const contractAbi = [{ 'constant':false, 'inputs':[], 'name':'withdraw', 'outputs':[], 'payable':false, 'stateMutability':'nonpayable', 'type':'function' }, { 'constant':true, 'inputs':[], 'name':'
const contractCode = '606060405260006002553415610014576000080fd5b6040516108823803806108828339810160405280805190910190506000818051610042929160200190610064565b505060018054600160a060020a03191633600160a0

const contractId = 'testId'; // contract id
const gas = '750000'; // insane hight gas price to make the demo work fast ;)

// instead of having a ABI and bitecode, we could read the contract of the FS (or have an input field with it)
// then we could live compile it (https://gist.github.com/tomconte/4edb83cf505f1e7faf172b9252fff9bf)
// we could even use Monaco (the editor, that powers Visual Studio Code), as WYSIWYG Editor
// https://github.com/Microsoft/monaco-editor and use a solidity plugin (demo: https://microsoft.github.io/monaco-editor/)
// then we could have our own remix.etherum.org and not rely on them... just saying...

const web3Api = new window.Web3(window.web3.currentProvider);
const contract = web3Api.eth.contract(contractAbi);
contract.new('"' + contractId + '"', {
  from: web3.eth.accounts[0], // use current MetaMask account
  data: contractCode, // use the contract bytecode
  gas: gas,
}, function (e, contractResponse) {
  if (typeof contractResponse === 'object') {
    if (typeof contractResponse.transactionHash === 'undefined') {
      swal('Ooops...', 'Error no transaction hash!', 'error');
    } else {
      swal('Nice!', 'Contract id: ' + contractResponse.address, 'success');
    }
  } else {
    swal('Ooops...', 'There was an error!', 'error');
  }
});
```