Integration of a Secure Physical Element as a Trusted Oracle in a Hyperledger Blockchain

Andreas Schaad[1], Tobias Reski[1] and Oliver Winzenried[2]
[1]*Faculty of Media, University of Applied Sciences Offenburg, Badstraße 24, 77652 Offenburg, Germany*
[2]*Wibu-Systems AG, Rueppurer Strasse 52, 76137 Karlsruhe, Germany*

Keywords: Software Protection, Blockchain, Hyperledger.

Abstract: Blockchain frameworks enable the immutable storage of data. A still open practical question is the so called "oracle" problem, i.e. the way how real world data is actually transferred into and out of a blockchain while preserving its integrity. We present a case study that demonstrates how to use an existing industrial strength secure element for cryptographic software protection (Wibu CmDongle / the "dongle") to function as such a hardware-based oracle for the Hyperledger blockchain framework. Our scenario is that of a dentist having leased a 3D printer. This printer is initially supplied with an amount of x printing units. With each print action the local unit counter on the attached dongle is decreased and in parallel a unit counter is maintained in the Hyperledger-based blockchain. Once a threshold is met, the printer will stop working (by means of the cryptographically protected invocation of the local print method). The blockchain is configured in such a way that chaincode is executed to increase the units again automatically (and essentially trigger any payment processes). Once this has happened, the new unit counter value will be passed from the blockchain to the local dongle and thus allow for further execution of print jobs.

1 INTRODUCTION

Blockchain frameworks enable the immutable storage of data (Yaga et al., 2018, Androulaki et al., 2018). A still open practical question is the so called "oracle" problem, i.e. the way how real world data is actually transferred into and out of a blockchain (Buck, 2017) while preserving its integrity. We present a case study that demonstrates how to use an existing industrial strength secure element for cryptographic software protection (Wibu CmDongle / the "dongle") to function as such a hardware-based oracle (Apla, 2019) for the Hyperledger blockchain framework.

Hyperledger is an open-source framework of blockchain technologies (Androulaki et al., 2018). It is a so-called permissioned network, where participants are known and have been provided with an identity when joining the network. This allows to support more efficient proof of work concepts than in traditional blockchain frameworks such as Bitcoin (Antonopoulus, 2017).

The Wibu CmDongle is a secure element (a "dongle") to enable cryptographic software protection and licensing of functionality. It can be attached to a device as a USB token or is integrated into an embedded system (Wibu, 2019). Recently a cloud-based software protection solution has been presented (Schaad et al., 2018).

Using the capabilities of the dongle, software can be cryptographically protected at run-time as fine-grained as controlling access to individual methods. A certificate chain rooted at the software vendor controls which customer should have access to which type of functionality (i.e. real-time decryption of code). This setup also supports commercial licensing where the same software is shipped but will be differently used.

The dongle consists of hardened cryptographic hardware as well as allows to persist data. One standard use case for such data are so called unit counters that allow to measure how often a certain action has been performed or which threshold a data value may have reached.

We present a case study that demonstrates how to use this industrial strength secure element for cryptographic software protection (Wibu CmDongle / the "dongle") to function as a hardware-based oracle (Apla, 2019) for the Hyperledger blockchain framework.

Because the invocation of any Hyperledger APIs is part of the protected software, we can achieve a

Schaad, A., Reski, T. and Winzenried, O.
Integration of a Secure Physical Element as a Trusted Oracle in a Hyperledger Blockchain.
DOI: 10.5220/0007957104980503
In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications (ICETE 2019)*, pages 498-503
ISBN: 978-989-758-378-0

Figure 1: Blockchain-based value chain (with Dongle).

seamless and tamperproof recording of real-world activities.

2 SCENARIO

2.1 Business Case

The real-world scenario we support is that of a leased industrial 3D printer limited to print only a certain amount of items (for example for printing dental inlays directly at a dentist's workplace). With every print action the unit counter of the attached dongle is decremented. As soon as a 0 threshold is reached the printer will not print anymore until the lessee (e.g. dentist) acquires more print units from the machines actual owner via an online portal to obtain an activation code that will reset the unit counter. In parallel the dentist has to order printing material from another vendor. The reasons why a blockchain may make sense in this context are based on the decision criteria recommended by NIST (Yaga et al., 2018):

- We need shared consistent data storage between participants
- More than one entity needs to contribute data (dentist, machine owner, material supplier)
- We only need Create and Read but no Update or Delete actions
- No PII data is required to be stored
- Our participants have a common economic interest but limited trust
- Any data storage needs to be immutable

More specifically, we were interested in the ability to execute and synchronize business logic between the participants.

We thus considered using a blockchain framework for:

- Storing unit counter data
- Defining domain logic (smart contracts) that automates transactions between participants in a value chain

Figure 2: 3 Intel NUCs with one attached Dongle.

As a suitable blockchain framework we identified Hyperledger (Androulaki et al., 2018) which supports:

- An operational model not based on a crypto currency
- A private consortium
- Lightweight consensus models
- Coordination of participant activity through smart contracts

2.2 Technical Setup

Our setup consists of several standard Intel NUCs (Intel NUC, 2018) to simulate the entities participating in our scenario. One of the NUCs has an attached dongle and represents the 3D printer. The reason why we chose the largest available form factor of a dongle is because of an integrated LED that allows us to visualize the current unit counter status.

The interaction between the protected software (e.g. the simulated 3D printer), the secure element and the Hyperledger blockchain is realized over several interfaces and follows the sequence in figure 3.

In a protected application an activity that leads to a unit decrement is invoked via the `CmAccess()` function in the application code. CodeMeter is a locally running background process that controls cryptographic access to cryptographically protected code. Any such access will request a `CmCrypt()` operation in the Dongle's firmware which will decrease the unit counter on the dongle by one (Figure 9).

Right after the `CmCrypt()` command we address the Hyperledger network and issue a `DecrementUsage` API call (this is done from the protected code (Figure 9) but could also be done from inside the secure element). We discuss the implementation of this API in the following section on our Hyperledger implementation (Figure 5).

This will lead to an immutable decrement of the

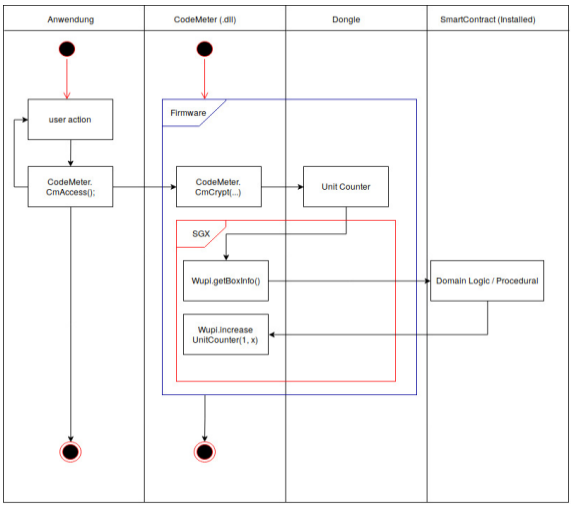

Figure 3: Architecture.

```
namespace org.example.mynetwork

asset Printer identified by printer_id {
    o String printer_id
    o String type
    o Integer usage
    --> Customer tenant
    --> Lender owner
    --> Ressourcer ressourcer
}
asset Debt identified by debt_id {…
}

abstract participant Network_Participant{
  o String name
  o String company
}

participant Customer identified by customer_id extends Network_Participant {…
}
participant Lender identified by lender_id extends Network_Participant {…
}
participant Ressourcer identified by ressourcer_id extends Network_Participant{…
}

transaction DecrementUsage {
  --> Printer printer
}
transaction Refill {…
}

event Refill_Status {…
}
```

Figure 4: Hyperledger Data Model.

current unit counter value we defined in the blockchain (also defined in the following section). More precisely, a series of transactions allows to determine the current (world) state at any time.

Regarding the LED within the dongle, we use green to indicate that sufficient units are available, orange for a currently ongoing decrement operation and red to indicate that no more units are available.

3 DESIGN AND IMPLEMENTATION

3.1 Data Model

Our data model is rather simplistic as defined in figure 4, but suffices to model our value chain (Figure 1). We distinguish between:

- asset
- participant
- transaction

Assets are modified via transactions and represent our device that works on basis of a unit counter. A participant takes part in the overall value chain. Here "o" refers to an attribute, while "-->" refers to an object.

The usage attribute of the printer defines the current available units and reflects the unit counter of the CmDongle. The Debt asset is used to reflect that any "refill" of print units will create a relationship between Customer, Lender and Ressourcer.

Customer in this context is our dentist, lender is the owner of the machine and Ressourcer is the material supplier.

```
async function decrementUsage(decUsage) {
    decUsage.printer.usage--;
    updateAsset('Printer',decUsage.printer);
}

async function updateAsset(type,data){…
}
async function addAsset(asset,type){…
}
function createAsset(type,key){…
}
async function status(printer) {…
}
async function refillPrinter(refill) {…
}
```

Figure 5: Hyperledger Chaincode.

The actual implementation of, for example, the `DecrementUsage` operation is done by chaincode in Hyperledger (Figure 5). Within our Hyperledger setup we used standard JavaScript to implement the domain logic. This is called chaincode and defines the actions that are executed when an external command from our printer application is received.

We defined explicit permissions in an access control list using these entities (Figure 6).

```
rule Printer_Transaction_Decrement {
    description: "Only customer can invoke transactions for printer"
    participant(p): "org.example.mynetwork.Customer"
    operation: READ, CREATE, UPDATE
    resource(r): "org.example.mynetwork.Printer"
    transaction(tx): "org.example.mynetwork.DecrementUsage"
    condition: (r.tenant.getIdentifier() == p.getIdentifier())
    action: ALLOW
}
```

Figure 6: Hyperledger ACLs.

4 DEMONSTRATION

In a first step we imagine that the printer is currently equipped with 5 print units. The local CmDongle reflects this in an immutable fashion (Figure 8). The same value initial value is maintained in the Blockchain when querying the Printer asset interface. We now invoke the `DecrementUsage` API call (Figure 5) from our simulated printer software. More precisely, through the integration of Wibu CodeMeter daemon any print action will enforce such a decrement at the local unit counter as well as in the Blockchain. Enforcement implies real-time cryptographic enforcement locally as well as in the Blockchain.

The effect of such a decrement operation can again be observed locally or in the Blockchain (Figure 8).

Once the unit counter hits a zero value, we invoke a refill operation (Figure 7) via a Python script that calls the corresponding chaincode.

```
#! /usr/bin/python
import requests
import sys
import uuid

debt_id = uuid.uuid4().hex
r = requests.post('http://192.168.0.102:3001/api/DecrementUsage', json = {
  "$class": "org.example.mynetwork.Refill",
  "printer": "resource:org.example.mynetwork.Printer#1",
  "amount": 10,
  "debt_id" : debt_id})
print(r.status_code)
if(r.status_code == 200):
    sys.exit(0)
else:
    sys.exit(-1)
```

Figure 7: Hyperledger API call (in Python).

This will now lead to creation of a Debt transaction which reflects that between the three participants a certain monetary value has been created. At the same time the local unit counter was increased by the requested amount (Figure 5).

The overall code is available on Github: https://github.com/AdSHSO/CmDongleHyperledger

5 SUMMARY

We have provided an integration of a Hyperledger blockchain with the Wibu CmDongle as an external hardware-based oracle. The dongle enforces cryptographic access control down to the individual method- or function level of an application through a local daemon.

Our scenario was that of a Dentist having leased a 3D printer. This printer is initially supplied with an amount of x printing units. With each print action the local unit counter on the Wibu CmDongle is decreased and in parallel a unit counter is maintained in the Hyperledger-based blockchain. Once a threshold is met, the printer will stop working (by means of the cryptographically protected invocation of the local print method).

Because the invocation of any Hyperledger APIs is part of the protected printer software, we can achieve a seamless and tamperproof recording of real-world activities.

The blockchain is configured in such a way that chaincode is executed to increase the units again automatically (and essentially trigger any payment processes). Once this has happened, the new unit counter value will be passed from the blockchain to the local dongle and thus allow for further execution of print jobs.

As a side-effect, our dongle can record any real-world activities even in a temporary offline scenario and later persist them to the blockchain.

Of course, this is only a basic proof of concept with the intent to demonstrate how a local secure element (the Wibu CmDongle) could assist as a trusted hardware oracle for a blockchain. We have not yet registered the secure element as a trusted blockchain participant, but this should only be minor technical issue when using Hyperledger as a framework and its existing identity management functionality.

We have also not fully provided an end to end scenario for processing payment information and resetting any unit counters. Again, this is seen as an engineering exercise as such an integration of payment solutions already does exist as part of the current Wibu-Systems technology stack. Likewise, we only implemented one example where chaincode invokes other chaincode (refill operation creates Debt).

We are not concerned with any throughput or performance issues as the figures reported in (Androulaki et al., 2019) appear to make Hyperledger suitable for real-time applications opposed to the consensus model in Bitcoin (Antonopoulus, 2017).

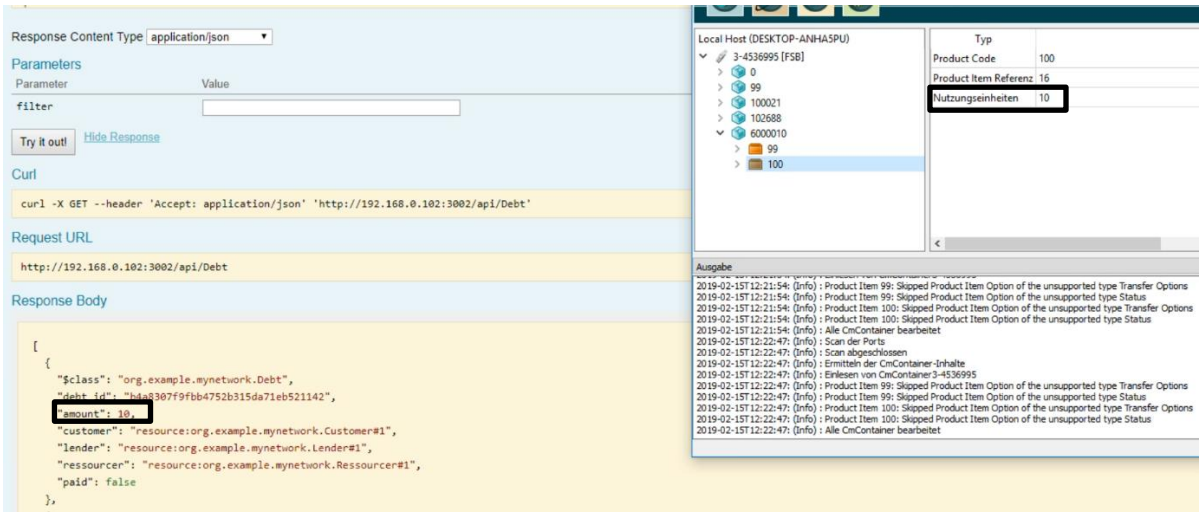

Figure 8: Local ("dongle") unit counter and blockchain unit counter.

Finally, as indicated in Figure 3, we currently investigate how to use Intel's SGX (Software Guard Extensions) in a next step to further increase the trust regarding the communication of the local dongle with the Hyperledger blockchain. That means that any Hyperledger API call from the local dongle should be invoked from within an enclave.

```
141         CodeMeter.CMACCESS2 cmAcc = new CodeMeter.CMACCESS2();
142         cmAcc.credential = cmCred;
143         cmAcc.ctrl |= CodeMeter.CM_ACCESS_NOUSERLIMIT;
144         cmAcc.firmCode = 6000010;
145         cmAcc.productCode = 100;
146         cmAcc.productItemReference = 16;
147         long hcmse1;
148         hcmse1 = CodeMeter.cmAccess2(CodeMeter.CM_ACCESS_LOCAL, cmAcc);
149
150         int error = CodeMeter.cmGetLastErrorCode();
151         if ( error == 32)
152         {…
163         else
164         {
165             CodeMeter.CMBASECRYPT2 cmBaseCrypt = new CodeMeter.CMBASECRYPT2();
166
167             cmBaseCrypt.ctrl |= CodeMeter.CM_CRYPT_FIRMKEY;
168             cmBaseCrypt.ctrl |= CodeMeter.CM_CRYPT_AES;
169             cmBaseCrypt.encryptionCodeOptions |= 1;
170             cmBaseCrypt.encryptionCodeOptions |= CodeMeter.CM_CRYPT_UCCHECK;
171             cmBaseCrypt.encryptionCodeOptions |= CodeMeter.CM_CRYPT_ATCHECK;
172             cmBaseCrypt.encryptionCodeOptions |= CodeMeter.CM_CRYPT_ETCHECK;
173             cmBaseCrypt.encryptionCodeOptions |= CodeMeter.CM_CRYPT_SAUNLIMITED;
174             CodeMeter.CMCRYPT2 cmCrypt = new CodeMeter.CMCRYPT2();
175             cmCrypt.cmBaseCrypt = cmBaseCrypt;
176             byte [] initkey = {…
182             System.arraycopy(initkey, 0, cmCrypt.initKey, 0, CodeMeter.CM_BLOCK_SIZE);
183
184             byte [] pbDest = {…
190             int res = CodeMeter.cmCrypt2(hcmse1,
191                                          CodeMeter.CM_CRYPT_DIRECT_ENC,
192                                          cmCrypt, pbDest);
```

Figure 9: Example protected Java code.

6 CONCLUSION

We provided a practical answer to the „Oracle" problem by using an existing industrial strength secure element as a hardware-based oracle. The main advantage of our work is that any API invocation of the Hyperledger blockchain is part of the cryptographically protected code. This implies that the vendor of the software can "hardwire" desired interactions with the blockchain and any man-at-the-end attacks (i.e. by the paying customer) are not possible (Figure 9). The vendor can protect functionality at build time of his software as fine-grained as at the individual method / function level and thus place any corresponding blockchain API calls where required. As each later customer has a unique ID this can then lead to individual API calls.

ACKNOWLEDGEMENTS

This work has been funded by a research cooperation agreement between University of Offenburg and Wibu-Systems AG. Alvaro Forrero, Thomas Falk and Alexander Eger did help with implementing the proof of concept.

REFERENCES

Androulaki et al., 2018*: Hyperledger Fabric: A distributed operating system for permissioned blockchains.* ACM Eurosys, 2018

Antonopoulus, A., 2017: *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. OReilly, 2017

Apla, 2019 *What is a blockchain oracle?* Available at: https://blog.apla.io/what-is-a-blockchain-oracle-2ccca433c026

Buck, J. 2017: *Blockchain Oracles, Explained.* Cointelegraph, Available at: https://cointelegraph.com/explained/blockchain-oracles-explained

Intel NUC, 2018 Available at: https://www.intel.de/content/www/de/de/products/boards-kits/nuc.html

Schaad et al., 2018 *Towards a Cloud-based System for Software Protection and Licensing*. ICETE (2) 2018: 698-702

Wibu, 2019: Wibu Systems AG Available at www.wibu.com

Yaga et al., 2018 *NISTIR 8202. Blockchain Technology Overview*. Available at: https://csrc.nist.gov/publications/detail/nistir/8202/final