Creating a TPM based smart contract for the Medical Supply Chain in Hyperledger Fabric

Student: Kevin Nanhekhan (k.r.nanhekhan@student.tudelft.nl), Responsible Professor/Supervisor: Kaitai Liang (katai.liang@tudelft.nl)

TU Delft

1. Research aim and questions:

There are critical concerns on blockchain regarding security and privacy of data. Securing the smart contracts run in blockchain with the use of trusted hardware is a one of the solutions. TPM[1] was chosen as here trusted hardware to focus on.

Main and sub-questions:

How to use the trusted hardware TPM to protect the execution of smart contracts?

- How does the execution of a smart contract within a ledger work?
- What are the different functionalities of a TPM?
- How can the functionalities of a TPM be integrated with a smart contract?
- How can the performance of the secure smart contract be assessed?

2. Methodology

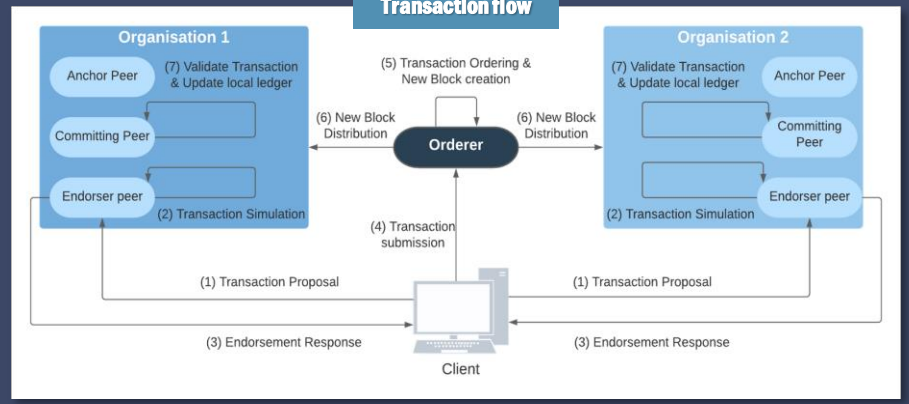
Search terms:
Smart Contracts, Trusted hardware, TPM, Trusted computing

Literature (2005 – 2021 period) from research databases:
Springer, Science Direct, IEEE, ACM

Coding Tools used:
The virtual machine **Hyper-V**[2] with TPM enabled.
Hyperledger Fabric[3] blockchain framework
Programming language **Go**
Opensource library **Go-TPM**[4] to interact with the TPM

Benchmark tools:
Hyperledger Caliper[5] and **Revive^cc**[6]

Transaction flow



3. Background

Hyperledger Fabric has various components

TPM has three main functionality[7]:
Secure storage
Platform measurement and reporting.
Platform authentication.

Pseudocode of the TPM function

```
1: function TPMHASH(input)
2:     rwc, err = open tpm2 using the go-tpm libary on the /dev/tpm0 path
3:     if error ≠ NULL then
4:         return error that tpm couldn't be opened
5:     end if
6:     dataToHash = convert input string to byte array
7:     hashdigest, hashError = calling the tpm2 hash function on the dataToHash
8:     if hashError ≠ NULL then
9:         return error occurred while hashing
10:    end if
11:    return hashDigest
12: end function
```

Performance results

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
Issue-function	1000	0	75.9	2.56	0.10	0.95	65.6
Request-function	15165	0	256.9	0.23	0.01	0.02	256.8
CancelRequest-function	16677	0	282.5	0.14	0.01	0.02	282.5
ApproveRequest-function	15216	0	257.8	0.14	0.01	0.02	257.8
CheckUserHistory-function	189	0	3.2	2.89	0.63	1.96	3.1
RejectRequest-function	18495	0	313.3	0.12	0.01	0.02	313.2
CheckAvailableMedicine-function	65	0	1.1	9.80	0.52	6.08	1.1
CheckRequestedMedicine-function	189	0	3.1	3.01	0.58	1.97	3.0
CheckHistory-function	173	0	2.8	2.89	0.64	2.08	2.8
SearchMedicineByName-function	1143	0	19.2	0.61	0.08	0.31	19.2
ChangeStatus-function	16808	0	284.7	0.10	0.01	0.02	284.7
CheckHolder-function	17710	0	300.0	0.10	0.01	0.02	300.0

Working Transaction

```
krm@vm:~/medical-supply/customers/application$ go run app.go
2022/01/21 22:00:06 ============ Sucessfully populated wallet ============
2022/01/21 22:00:06 TPM Key used is: n[N6/e@@@R@@@@k@@@@
2022/01/21 22:00:06 Choose number to invoke function:
1 - Request a medicine
2 - Cancel request
3 - Check User History
4 - Search Medicine by name
5 - Check available medicine
1
2022/01/21 22:00:08 Medicine name (e.g. Aspirin):
Aspirin
2022/01/21 22:00:12 Medicine number (e.g. 00001):
00001
2022/01/21 22:00:14 --> Submit Transaction: Request, function sends request for medicine.
2022/01/21 22:00:17 {
        "checkSum": "",
        "medName": "aspirin",
        "medNumber": "00001",
        "disease": "pain management",
        "expiration": "2022.05.09",
        "price": "$10",
        "holder": "alice",
        "currentState": 2,
        "class": "org.medstore.medicalsupply",
        "key": "MedStore:aspirin:00001"
}
```

4. Prototype:

For the application, a simplified Medical Supply chain will be created.
MedStore: Pharmacy network providing medical supply.
Medical Supply: Asset which is traded on the network.
Customers: Organization 1 which orders medicine from the MedStore.
Regulators: Organization 2 handling customers requests.

Integrate TPM to mitigate the security limitation: **Wormhole attack**
TPM solution:
Checksum: Check no alterations to Medical supply
Anonymity: User provided names are hashed
Authentication: TPM-generated key and hashed username stored

5. Conclusion

Working prototype which shows potential as TPM has been used to secure a Medical application. However, security could not be fully measured.

Future Work:
- Research other use cases for trusted hardware
- Development of more security analysis tools

References:

[1] Trusted Computing Group (TCG). Trusted platform module, 2022. Available: https://trustedcomputinggroup.org/work-groups/trusted-platform-module/
[2]Introduction to Hyper-V on windows 10., 2022. Available: https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/
[3] The Linux Foundation. Hyperledger fabric main documentation., 2022. Available: https://hyperledger-fabric.readthedocs.io/en/latest/whatis.html
[4] Go-tpm module, 2021. Available: https://pkg.go.dev/github.com/google/go-tpm.
[5] The Linux Foundation. Hyperledger caliper,2022. Available: https://hyperledger.github.io/caliper/.
[6] sivachokkapu. Revive cc, 2020. Available: https://github.com/sivachokkapu/revive-cc
[7] Kenneth Ezirim, Wai Khoo, George Koumantaris, Raymond Law, and Irippuge Perera. Trusted platform module – a survey. 11 2012

Full code available at: https://github.com/Kevin-RN/medical-supply