decode

Implementation of Blockchain platform and ABC in DECODE pilots

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 732546

Project no. 732546

DECODE

DEcentralised Citizens Owned Data Ecosystem

D3.10 Implementation of Blockchain platform and ABC in DECODE Pilots

Version Number: V1.0

Lead beneficiary: DYNE.ORG

Due Date: 30/06/19

Author(s): Denis Roio, Puria Nafisi Azizi, Andrea D'Intino (DYNE.ORG)

Editors and reviewers: Jim Barrit (TW), Ioannis Psaras (UCL), Jaap-Henk Hoepman (RU)

Dissemination level:		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Approved by: Francesca Bria (DECODE project coordinator)

Date: 25/09/2019

This report is currently awaiting approval from the EC and cannot be not considered to be a final version.

1 Tables of Contents

1.Introduction

This deliverable consists of the implementation of a distributed ledger technology (DLT) that integrates with DECODE technologies, more specifically with the Zencode smart-rules language and consequently with the Zenroom virtual machine (VM) that executes them.

Zencode is a simple, non-touring complete, natural language interpreter based on a domain specific language (DSL) that can run and execute these transformations inside a very portable virtual machine (Zenroom) capable of cryptographic transformations (Roio, 2017). As of today Zenroom has been successfully adopted as a core component by every DECODE pilot.

The challenge of this deliverable is that of connecting the execution of Zencode to a blockchain (DLT) whose basic function is to verify that each operation can be reproduced, leading to the same output when fed with the same input. In other words, the addition of a DLT grants the provability of cryptographic operations, where deviance is handled by a consensus protocol.

More specifically, this deliverable details and demonstrates the execution of the Coconut crypto model adopted by DECODE pilots and in particular DECIDIM's pilot (DDDC). This demonstration suffices for the purpose of the deliverable since other pilots (in particular the Barcelona IoT pilot) which require only a subset of the functionalities implemented for Coconut petitions. For this reason, our work went not only in the direction of demonstrating use on the blockchain, but has also created conditions in which the same transformations can be easily executed as local or remote database operations.

In both cases it has been necessary to evolve the Zenroom implementation into a fully deterministic VM with no variable inputs. Particular effort was used into providing it with a reliable source of cryptographic pseudo-random generator, whose seed is derived from the cryptographic hash of previous blocks on the blockchain.

The relevant source code repositories for this deliverable are:

- https://github.com/DECODEproject/zenroom
- https://github.com/DECODEproject/zenroom-tp-python
- https://github.com/DECODEproject/zenroom-py
- https://github.com/DECODEproject/dddc-pilot-contracts
- https://github.com/DECODEproject/temet_nosce
- https://github.com/hyperledger/sawtooth-core
- https://github.com/dyne/docker-dyne-software (zenroom subdir)

2.Integration

Zenroom was thought and written from bottom up as a truly multiplatform application: being written in ANSI C99 with no external dependencies, porting to any standard software platform (Windows, Linux, Mac OSX, Android IoS) is straightforward.

2.1. Continuous integration

Zenroom is not a "vm" in that sense, it is in fact a "process virtual machine".

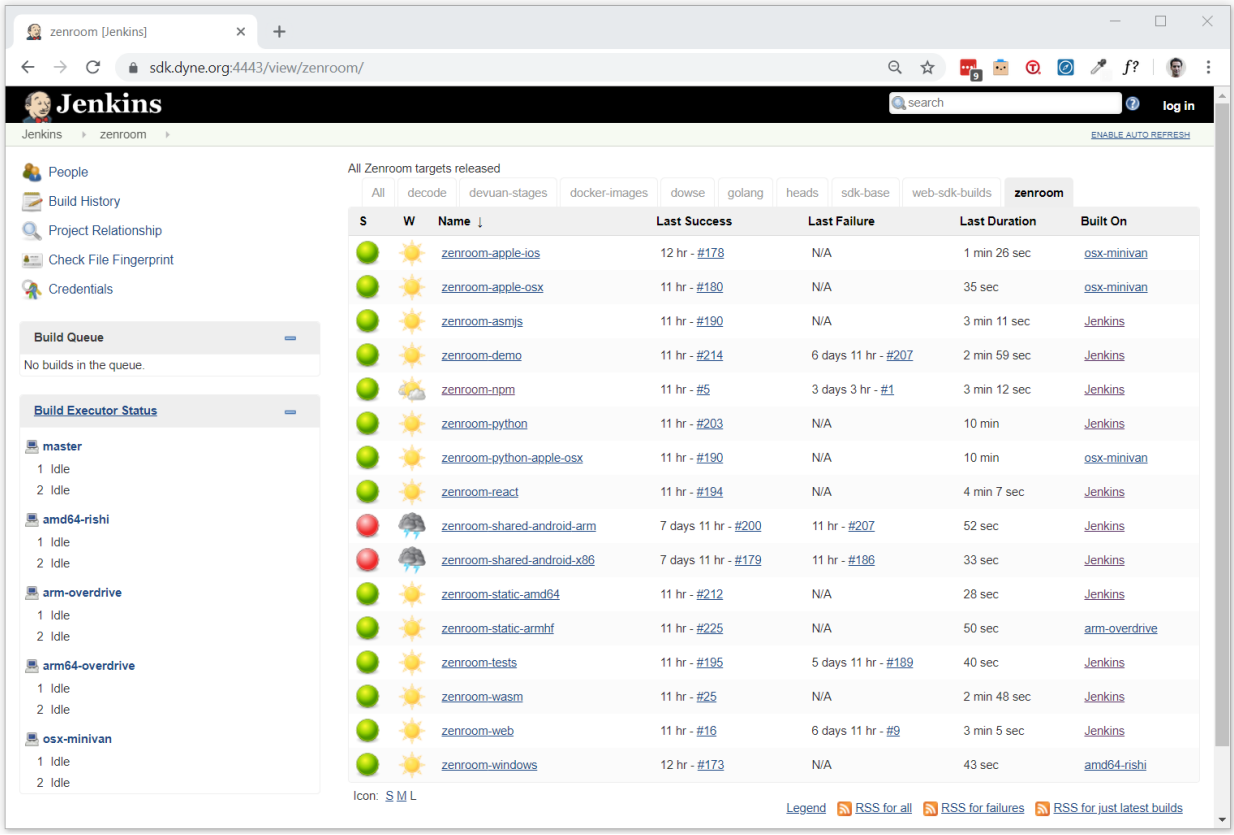
it aims to provide a platform-independent programming environment that abstracts away details of the underlying hardware or operating system and allows a program to execute in the same way on any platform.

Zenroom is in fact fully independent from its underlying OS, providing an high degree of determinism for complex cryptographic flows happening in multi-platform environments.

In order to make sure that the latest version of the Zenroom source code is constantly built for every chosen target platform, in a repeatable fashion (so called “continuous integration”), Dyne.org has setup and configured the widely used CI platform “Jenkins”, and developed scripts that:

- Build Zenroom once a day
- Execute automated tests for
    - Cryptography
    - Arithmetic
    - Zencode parser
    - Integration
    - Generic unit

Dyne.org’s Jenkins platform can be visited at https://sdk.dyne.org:4443



Each software developer involved in the development of DECODE pilots has been instructed to download the latest version of Zenroom built for his target platforms on a regular base. The current supported platforms are:

- For Linux:
  - Vanilla Linux AMD64 (for server applications)
  - Vanilla Linux ARMHF (for Arm boards like Raspberry Pi)
  - Linux with python binders
- For Windows AMD64
- For MacOS:
  - Vanilla Mac OSX
  - Mac OSX with python binders
- For mobile platforms:
  - For iOS
  - For Android:
    - Android ARM
    - Android X86
  - For “React Native”
- For Javascript (compiled using Webassembly and Emscripten)
  - ASMJS
  - Zenroom “Demo”, for the Zenroom demo web application
  - WASM
  - Generic Javascript build “Web”

2.2. Use in pilots

After extensive analysis in DECODE we assessed that only two are the pilots that can truly benefit from Attribute Based Credential system and the use of a blockchain to delegate the verification of claims to a decentralized network of peers. These are:

1. The Barcelona pilot "DDDC"
2. The Amsterdam pilot "18+"

The DDDC pilot has integrated the Decidim platform to share a petition with participants, sign it with a mobile app and count the signatures in a cryptographically secure and completely anonymous way. This pilot involves a several microservices, connected to an admin panel, communicating with a mobile app and storing data on a database and a blockchain. Its entry-point is on-line at https://dddc.decodeproject.eu

The work of multiple teams in the consortium lead to the successful integration of the Zenroom VM using Zencode contracts[1] and the Coconut[2] crypto scheme. The DECIDIM codebase has been adapted in the DDDC instance[3].

In order to realize end-to-end encryption for this system, a react-native mobile app[4] has been developed and integrated with the DDDC instance authentication system. The mobile app stores a participant's credential keypairs and receives signed credentials from an credential-issuer API[5]; using these credentials, the mobile app then and emits claims (credential proofs) and petition signatures (coconut petition scheme) that are sent to a petition API[6] that connects it to the blockchain implementation.

The blockchain implementation we developed is based on Hyperledger Sawtooth and Zenroom, is aptly called Sawroom[7] and is made with a custom transaction processor[8].

This pilot has been very useful to understand the strengths and weaknesses of our approach and, following the DDDC event demonstration, it prompted more development tasks: the current rewrite of the mobile app[9] and the rewrite of a transaction processor that is specific to the petition scenario[10].

---

[1] https://github.com/DECODEproject/dddc-pilot-contracts

[2] https://arxiv.org/pdf/1802.07344.pdf

[3] https://github.com/DECODEproject/DDDC-instance

[4] https://github.com/DECODEproject/decode-mobile-app

[5] https://github.com/DECODEproject/credential-issuer

[6] https://github.com/DECODEproject/dddc-petition-api

[7] https://github.com/DECODEproject/sawroom

[8] https://github.com/DECODEproject/sawtooth-tp-zenroom

[9] https://github.com/DECODEproject/decodev2

The Amsterdam pilot "18+" consists of a platform to cryptographically store users credentials (age in this case) allowing citizens to prove their age without the need to show their ID card, thus protecting their privacy from surveillance cameras (who may be recording pictures of the their ID cards) and undesired attention. This pilot includes passport scanners, as well as a network of microservices delivering and showing the credentials via mobile browsers.

Its implementation[11] consisted of a box capable of scanning European passports and cryptographically authenticate their contents, including the actual photo stored in JPG2000 and retrieved via NFC/RFID. It then uses Zenroom to run the ABC crypto and eventually store it on blockchain. The blockchain wiring hasn't been deployed as it turned as not strictly necessary for this pilot, is available in a simple form through the same system provided for the DDDC pilot.

The entry-point of the pilot is on-line at https://decode.amsterdam

[10] https://github.com/DECODEproject/petition-tp-python

[11] https://github.com/Amsterdam/decode_passport_scanner

3.Blockchain

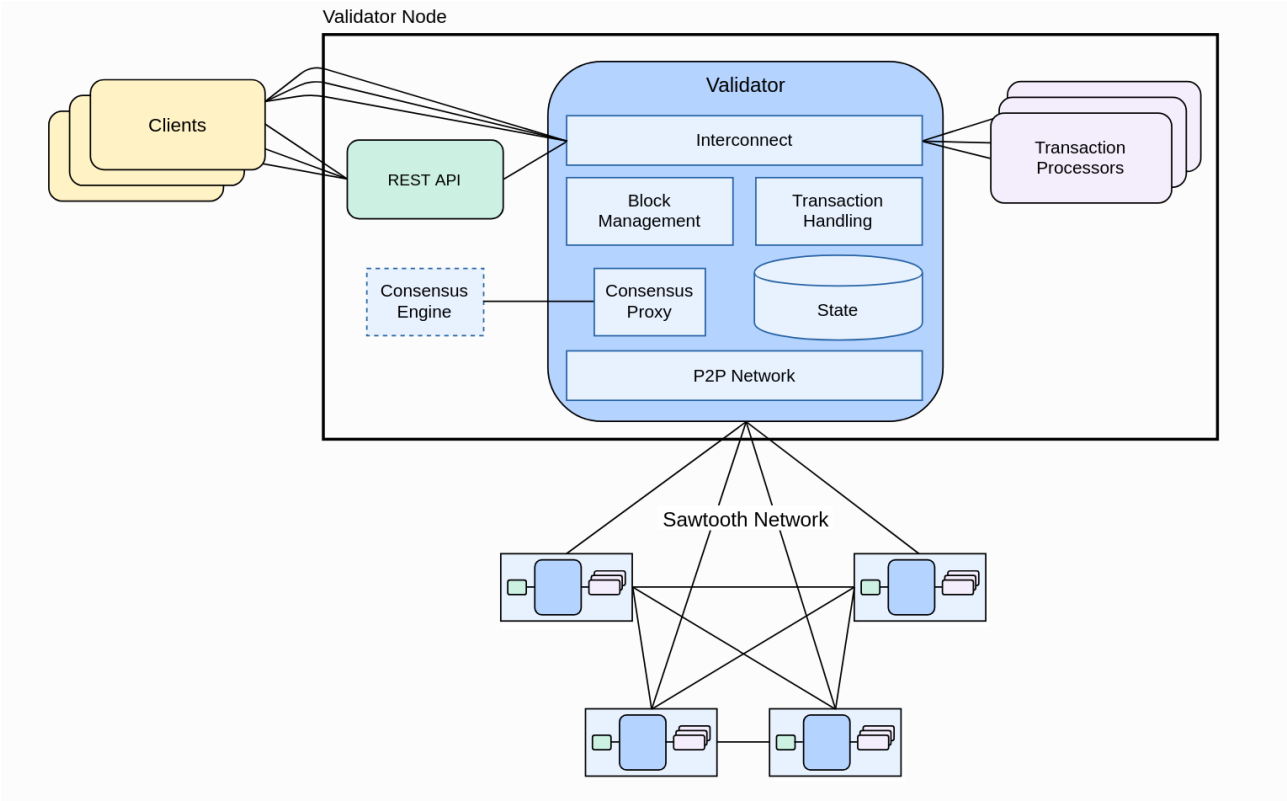
DECODE has benefited from research and development made on the Chainspace software implementation (Al-Bassam et al., 2017), which constituted an early lab test-bed (in-vitro). To fulfill the goals outlined by this document it has been necessary to adopt a working DLT implementation that substitutes this component in DECODE, mostly due to operational shortcomings related to its capacity to scale, its reliability and the maintainability of its code.

The DLT implementation chosen for production use in DECODE is Sawtooth (Olson et al., 2018) which is a free software part of the Hyperledger consortium coordinated by the Linux Foundation. The choice of Sawtooth is motivated by the fact that this blockchain has been engineered with simplicity and modularity in mind since the very beginning. Sawtooth does not impose any specific VM or consensus algorithm, allows building "permission-less[12]" as well permissioned[13] blockchain clusters and provides an extensively documented Application protocol interface (API) and an Software Development Kit (SDK) to plug custom made components. This approach absolves to a fundamental aim in DECODE: that of developing technologies that are blockchain agnostic and portable across different DLT platforms.

Within the current DLT panorama then Sawtooth proves itself to be well beyond infancy by providing a production-ready setup that can clearly demonstrate the functioning of our VM and the Coconut crypto contracts in a deterministic environment.

[12] "Permissionless" is a popular blockchain term indicating DLT setups that have no central authority, nor a federated group of authorities, but are open to include validating nodes without the need for an authorization and are ready to assess the honesty of notes based on a consensus algorithm.

[13] "Permissioned" indicates a closed network of nodes that share a distributed ledger whose validating operations can be only executed on authorized devices.



Zenroom for Sawtooth is basically plugging in as two components:

1. Client to send queries to the REST API
2. Transaction Processor (TP) to confirm deterministic execution to the Validator

The implementation is written in Python v3 and also in this case it is kept very minimal: the TP is executing only Zencode scripts (BDD[14]) which are locking down the execution of incoming commands to a strict DSL whose security will improve through next iterations. The Client is just redirecting the output (or "piping") from stdin to Sawtooth's REST API any text, but of course only Zencode will be accepted, anything else will return an error. Both successfully parsed Zencode which doesn't lead to deterministic results (when executed multiple times, it leads to different results) as well as error messages, are not saved on the blockchain.

To facilitate testing a docker compose script:

```
wget https://raw.githubusercontent.com/dyne/docker-dyne-software/master/zenroom/sawtooth-docker-compose.yaml > decode-zenroom-sawtooth.yaml
docker-compose -f  decode-zenroom-sawtooth.yaml up
```

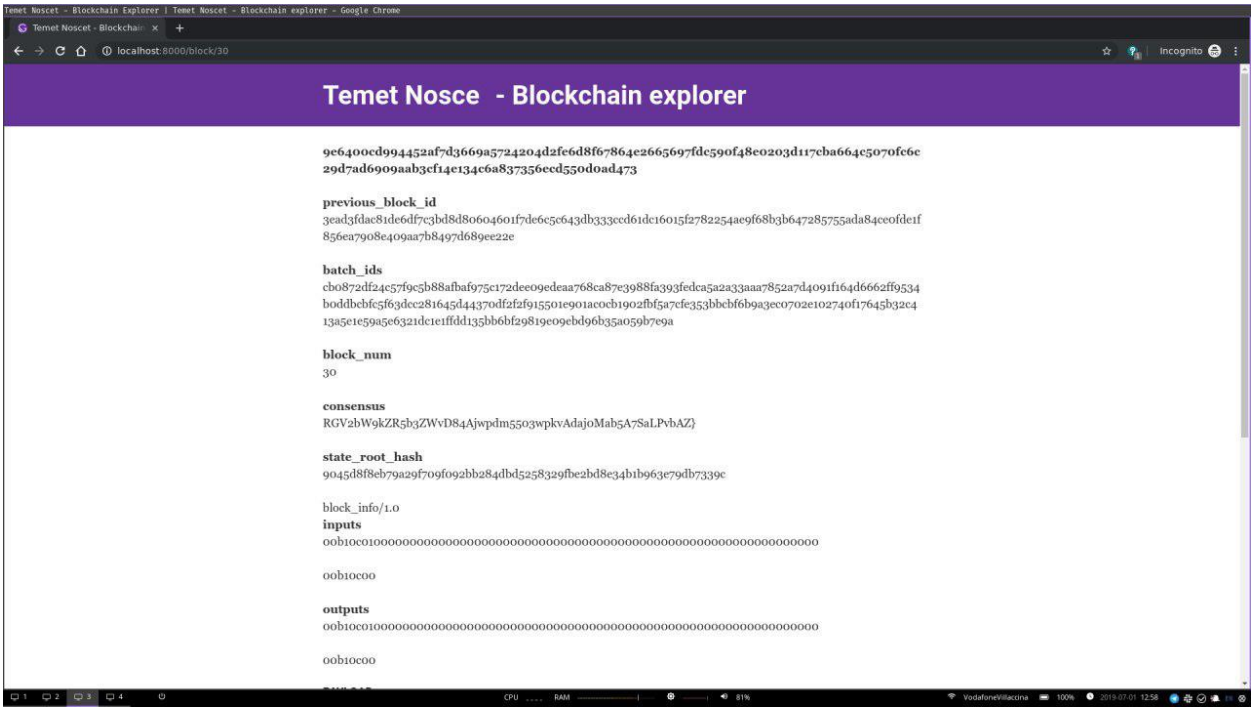
The TP implementation is here: https://github.com/DECODEproject/zenroom-tp-python

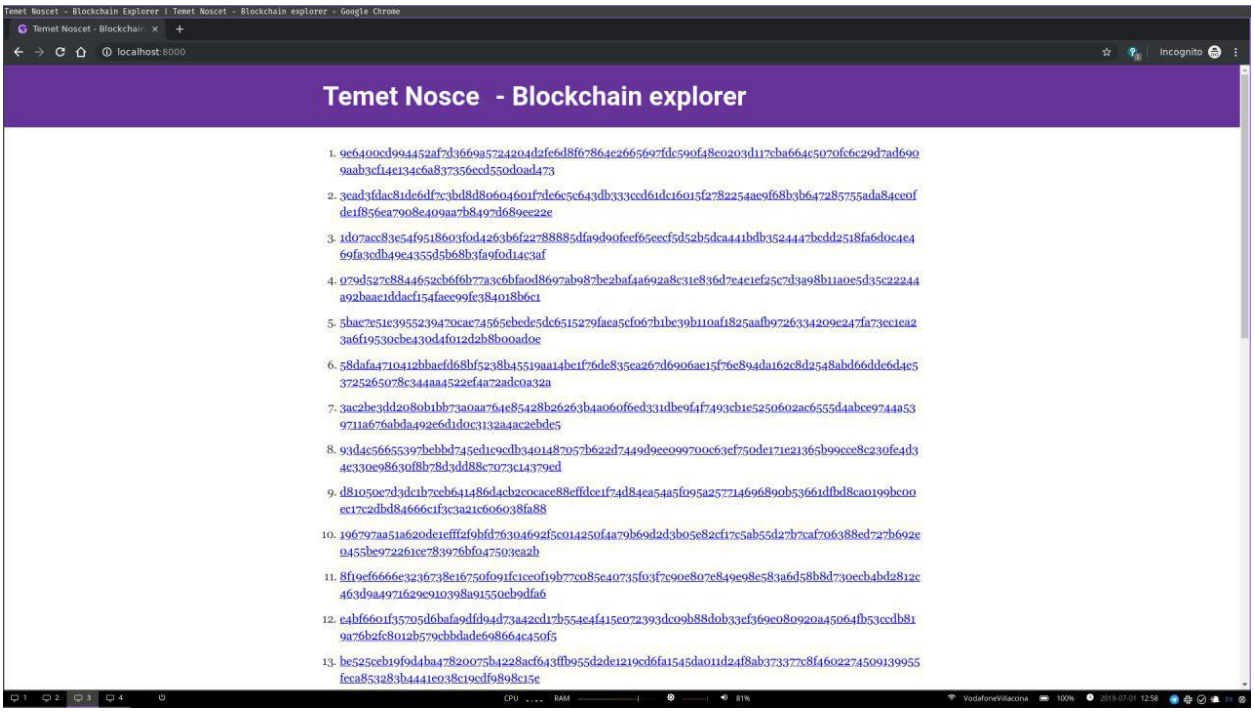
To facilitate the demonstration, we have also built a minimalist blockchain-explorer that is capable of executing Graph Query Language (GraphQL) operations on top Sawtooth and that can be easily customised. The name of this

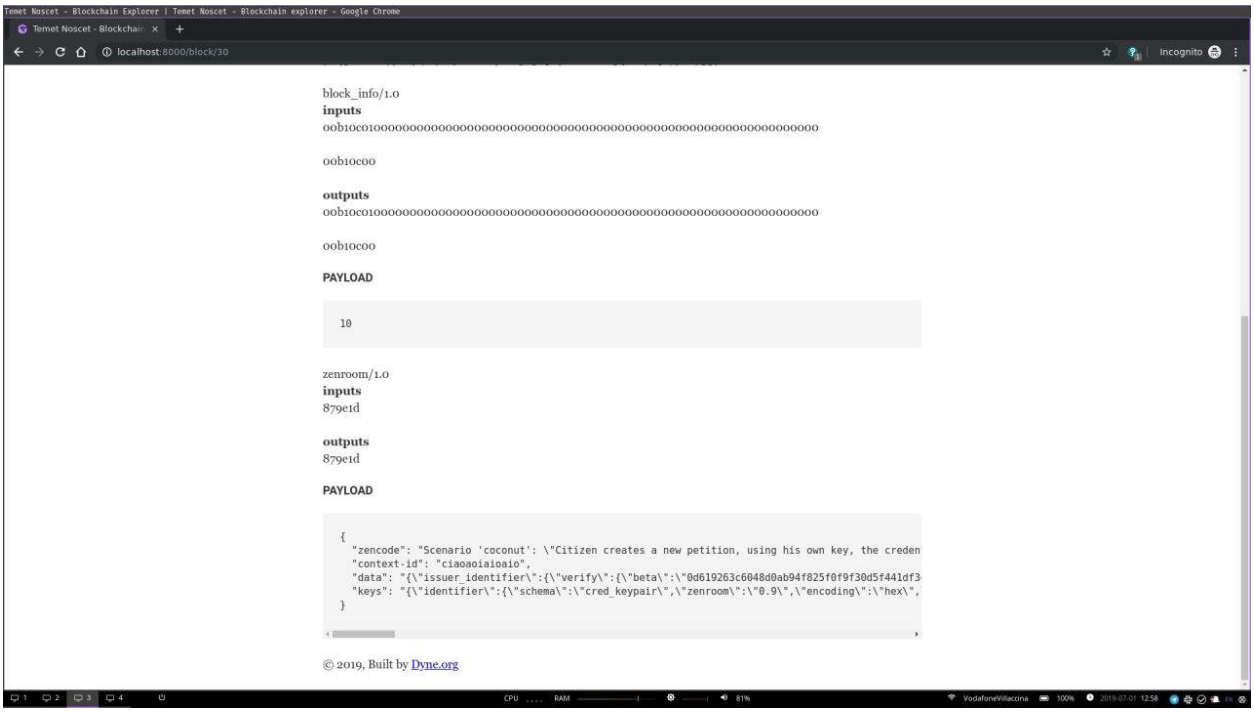
[14] Behaviour Driven Development, see bibliography Soeken, M., Wille, R., Drechsler, R.

new DECODE software is "Temet Nosce" and it constitutes just a minimal implementation mostly useful for lab tests. The most notable achievement of Temet Nosce is that of verifying the usefulness of GraphQL use in the implementation of a block explorer, an approach that has notably reduced the number of lines of code needed for its implementation hence lowering its complexity. Temet Nosce source:

https://github.com/DECODEproject/temet-nosce and screenshots below:




Temet Noscet - Blockchain explorer | Temet Noscet - Blockchain explorer - Google Chrome
Temet Noscet - Blockchain
localhost:8000
Incognito
Temet Nosce - Blockchain explorer
1. 9e6400cd994452af7d3669a57242o4d2fe6d8f67864e2665697fdc590f48e0203d117cba664c5070fc6c29d7ad6909aab3cf14e134c6a837356ecd550d0ad473
2. 3cad3fdac81de6df7c3bd8d806046o1f7de6c5c643db333ccd61dc16015f2782254ae9f68b3b647285755ada84ce0fde1f856ea7908e409aa7b8497d689ee22e
3. 1d07acc83e54f9518603f0d4263b6f2278885dfa9d90feef65eecf5d52b5dca441bdb3524447bcdd2518fa6d0c4e469fa3cdb49e4355d5b68b3fa9f0d14c3af
4. 979d527c8844652cb6f6b773c6bfa0d8697ab987be2baf4a692a8c31e836d7e4e1ef25c7d3a98b11a0e5d35c22244a92baae1ddacf154faee99fe384018b6c1
5. 5bac7e51e395523947ocac74565ebede5dc6515279faca5cf067b1bc39b110af1825aafb97263342o9e247fa73ec1ea23a6f19530cbe430d4f012d2b8b00ad0e
6. 58dafa4710412bbaefd68bf5238b45519aa14be1f76de835ea267d6906ae15f76e894da162c8d2548abd66dde6d4e53725265078c344aa4522ef4a72adc0a32a
7. 3ac2be3dd2080b1bb73a0aa764e85428b26263b4a060f6ed331dbe9f4f7493cb1e5250602ac6555d4abce9744a539711a676abda492e6d1d0c3132a4ac2ebde5
8. 93d4c56655397bebbd745ed1c9cdb3401487057b622d7449d9ee099700c63ef750de171e21365b99cce8c230fe4d34c330e98630f8b78d3dd88c7073c14379ed
9. d81050e7d3dc1b7ceb641486d4cb2c0cace88effdce1f74d84ca54a5f095a25771469689ob53661dfbd8ca0199bc00ec17c2dbd84666c1f3c3a21c6066038fa88
10. 196797aa51a620de1efff2f9bfd76304692f5c014250f4a79b69d2d3b05e82cf17c5ab55d27b7caf706388ed727b692e0455be972261ce783976bf047503ea2b
11. 8f19ef6666e3236738e16750f091fc1ee0f19b77c085e40735f03f7c90e807e849e98e583a6d58b8d730ecb4bd2812c463d9a4971629e910398a91550cb9dfa6
12. e4bf6601f35705d6bafa9dfd94d73a42cd17b554e4f415e072393dc09b88d0b33ef369e080920a45064fb53ccdb819a76b2fc8012b579cbbdade698664c450f5
13. be525ceb19f9d4ba47820075b4228acf643ffb955d2de1219cd6fa1545da011d24f8ab373377c8f46022745091399559feca853283b4441e038c19cdf9898c15e


Temet Noscet - Blockchain explorer | Temet Noscet - Blockchain explorer - Google Chrome
Temet Noscet - Blockchain
localhost:8000/block/30
Incognito
block_info/1.0
inputs
00b10c0100000000000000000000000000000000000000000000000000000000000000000000
00b10c00
outputs
00b10c0100000000000000000000000000000000000000000000000000000000000000000000
00b10c00
PAYLOAD
10
zenroom/1.0
inputs
879e1d
outputs
879e1d
PAYLOAD
{
"zencode": "Scenario 'coconut': \"Citizen creates a new petition, using his own key, the creden
"context-id": "ciaoaoiaioaio",
"data": "{\"issuer_identifier\":{\"verify\":{\"beta\":\"0d619263c6048d0ab94f825f0f9f30d5f441df3
"keys": "{\"identifier\":{\"schema\":\"cred_keypair\",\"zenroom\":\"0.9\",\"encoding\":\"hex\",
}
© 2019, Built by Dyne.org

3.1. Licensing

Since Sawtooth's software is developed outside of the DECODE consortium, we need to dedicate some attention to details regarding its licensing scheme, falling under the Apache License which is a free and open source license contemplated for use in our project.

Within recent DECODE missions we have made personal contact with the Sawtooth developers and communicated them our intentions, also receiving some useful coaching in the context of a "Hyperledger hacking workshop" in Amsterdam. On the wave of enthusiasm for the initiative we have also considered the entry of Zenroom's software within the Hyperledger products consortium, effort that may be envisioned in the framework of DECODE's standardization activities. The idea however has been abandoned due to licensing incompatibilities between Zenroom (Dyne.org foundation copyright, licensed as Affero GPL) and Hyperledger components whose license needs to be Apache (APL) and whose copyright needs to be resigned entirely to the consortium.

Nevertheless the adoption of Sawtooth is fully compatible with our licensing schemes and while allowing us to rely on a well developed and maintained DLT implementation it does not impact the free and open source software licensing scheme of components in DECODE, since all interactions between the Zenroom VM and Sawtooth are based on network connections rather than binary linking at compile time or execution time.

4.Determinism

Making the Zenroom VM fully deterministic, didn't require a large effort, especially considering that the software has been designed ground up to work with DLTs; in particular Zenroom has already no access to the filesystem of the host, to the network of the host as well to sources of data providing variable information as for instance functions returning the current date and time. The missing tasks then were two:

1. Provide a way for host applications (the external calling process) to input a random seed in Zenroom, making sure that an equal random seed across different executions leads exactly to the same series of results.

2. Make sure that all ports of Zenroom for any existing target will lead to same consistent results, including hashes, de/coding of binary data and cryptographic transformations.

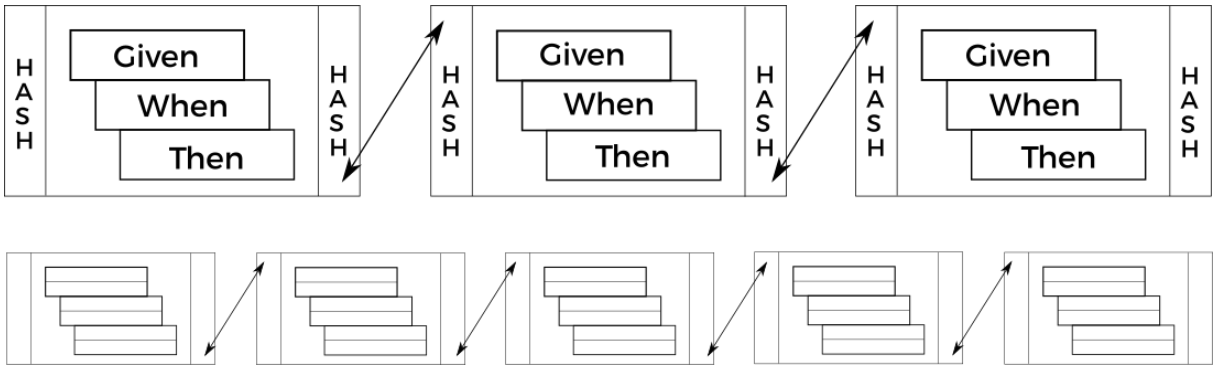
The first point has been solved by the implementation of a new API function in Zenroom:

```
// deterministic random calls for host applications providing their
// own seed for each use of the RNG class inside Zenroom
int zenroom_exec_rng_tobuf(...);
int zencode_exec_rng_tobuf(...);
```

The second point has been empirically verified both in lab tests (in-vitro) by continuous integration executed at every new code change to avoid regressions and on different platforms and targets; as well in-vivo by iterating through the different implementations of DECODE software which ended up being facilitated by the deterministic nature of Zenroom, a journey that is described in this new blog post "Cryptographic data integrity in a multiplatform environment" whose conclusion we highlight here:

> "[...] developers in DECODE could save a lot of time demanded by the necessity to align across different teams from multiple organizations, adopting different architectures, libraries and languages: the use of a VM like Zenroom has benefitted our workflow by lowering the friction and helping reach consistent results also across very complex cryptographic data transformations."

As a direct result of the "zenroom_exec_rng_tobuf" call used in our implementation of Sawtooth's TP, when running connected to a blockchain now Zenroom uses the hash of the previous block as a seed, making the whole chain of event deterministic and any node easily capable of reconstructing the validity with simple read-only access. This graphic may help explain at last:



In simple terms then our blockchain implementation consists of a generic system where each transaction is made of a Zencode script whose pseudo-random number generator (RNG) is sown with the hashes of the previous transaction, verifiable by any node with read-access to its blocks and capable of running a Zenroom VM.

5.Pragmatism

Under this brief section we intend to highlight an important research finding that emerges in this project, both from our experience with the Barcelona IoT pilot, as well from our collaboration with expert engineers at the European Commission: a blockchain is not always needed by projects willing to engage with verifiable computing and immutable data. To the contrary, a blockchain may be a burden where cryptographic transformations are only needed on GET/SET operations on a database that is replicated across multiple nodes. While being conscious this scenario does not realize a DLT implementation, we believed this aspect can be easily addressed by DECODE thanks to the modular architecture of our software and that it should be addressed since it still involves the realization of relevant pilots.

Therefore, what we call pragmatism here has been an implementation of Zenroom driven transformations done transparently using an in-memory, on-disk and replicated database. For this implementation we used the most popular key/value store across the ICT industry called Redis (Nelson, 2016) be it also for the aesthetic affinity about clean and minimalist code we share with its author Salvatore Sanfilippo.

This effort lead to the implementation of a Zenroom module for Redis which, once distributed by the final release of DECODE, will be an easy to install and deploy database that can seamlessly execute all the cryptographic transformations in Zenroom without the need to deploy a DLT, be it for the sake of testing, ease of integration with a DLT node or educational purposes.

The Zenroom redis module implementation resides inside Zenroom's source code and is made in two flavors, following the old SDKv1 specification and the ongoing SDKv2 development enhancing concurrency and performance: its target is built using 'make Linux-redis' and is undergoing testing and reviews.

We predict this new target will greatly facilitate the deployment of a replicable VM like Zenroom across cloud based micro-services, but also the familiarization of a growing number of web and front-end developers with the benefits offered by the Zencode DSL in conjunction with the simple and powerful functionality of Redis.

6.Future directions

DECODE's blockchain has now come to a point in which it is extremely convenient to merge other mature technologies and plug them in order to benefit of their development. This is especially the case with the Social Wallet API (SWAPI) developed in detail in the H2020 project Commonfare (Roio et al., 2015; Roio and Beneti, 2017), which can be now easily plugged to DECODE via its "freecoin-lib" component to benefit of its well tested, sanitized and scalable REST API as well of a growing number of client implementations, blockchain explorers and interactive scripting consoles. Work in this direction has already started within the current phase of DECODE.

Regarding the PRNG implementation for cryptographic random, Zenroom operates permutations on the seed leading to cryptographic robust random bytes using basic implementations for practical random generation in software (Viega, 2003) which we plan to enhance soon with a modern implementation of Fortuna (McEvoy et al., 2006).

The Redis module implementation is very promising and has also led to a bug fix which may be accepted as a contribution to this prestigious software. A low-hanging fruit in this sense will be the implementation of a standalone and ready to use cryptographic DB (simple key/value store) that executes cryptographic operations transparently.

At last Zenroom will be improved in order to simplify the Zencode language, document it and facilitate its test deployment on blockchain and in production across DECODE OS nodes: a 1.0 is predicted to be released by the end of the DECODE project with a stabilization of its API and the addition of more modern block ciphers and implemented contracts for tokens.

The Dyne.org development team is undergoing admission to the ISO standardization group TC 307 as well has applied to enter the Sovrin consortium and consequently all the developments above and future priorities will be informed by the wider horizons of community stakeholders at least for the core interoperability aspects, for which we foresee more blockchain integrations even beyond Hyperledger in the coming years.

7.Bibliography

- Al-Bassam, M., Sonnino, A., Bano, S., Hrycyszyn, D., Danezis, G., 2017. Chainspace: A sharded smart contracts platform. ArXiv Prepr. ArXiv170803778.
- McEvoy, R., Curran, J., Cotter, P., Murphy, C., 2006. Fortuna: cryptographically secure pseudo-random number generation in software and hardware.
- Nelson, J., 2016. Mastering Redis. Packt Publishing Ltd.
- Olson, K., Bowman, M., Mitchell, J., Amundson, S., Middleton, D., Montgomery, C., 2018. Sawtooth: An Introduction. Linux Found. Jan.
- Roio, D., 2017. Data Privacy and Smart Language requirements, its initial set of smart rules and related ontology.
- Roio, D., Beneti, A., 2017. REPUTATION DIGITAL CURRENCY AND NETWORK DYNAMICS COMPONENTS 18.
- Roio, D., Sachy, M., Lucarelli, S., Lietaer, B., Bria, F., 2015. Design of Social Digital Currency. -CENT.
- Viega, J., 2003. Practical random number generation in software, in: 19th Annual Computer Security Applications Conference, 2003. Proceedings. pp. 129–140.
- Soeken, M., Wille, R., Drechsler, R., 2012. Assisted behavior driven development using natural language processing, in: International Conference on Modelling Techniques and Tools for Computer Performance Evaluation. pp. 269–287.