EMITTER International Journal of Engineering Technology
Vol. 13, No. 1, June 2025, pp. 139~155
DOI : 10.24003/emitter.v13i1.949

A Detailed Set of Ideas for Designing a Quantum Computing Framework Based on Smart Contracts Configured Using Foundry and Qiskit

Alexandru-Gabriel Tudorache

Department of Computer Science and Engineering, Gheorghe Asachi Technical University of Iasi, Bd. Dimitrie Mangeron, Iasi, Romania
Corresponding Author: alexandru-gabriel.tudorache@academic.tuiasi.ro

Received January 30, 2025; Revised March 22, 2025; Accepted June 2, 2025

Abstract

The purpose of this paper is to describe a new system design for integrating quantum computing algorithms (and their results) into a blockchain network. In this selected context, we can use, create and upload smart contracts (SCs) that allow users to perform various quantum computations, by using the corresponding circuits. We are therefore proposing a system that uses gas fees in the blockchain context, in order to offer access to certain circuits and their simulation results; the system also allows for the previously analyzed circuits to become publicly available, through SCs – this can act like a quantum circuit encyclopedia. Most users in the first generation will have to pay, in addition to the normal transaction fees (gas) required to call the SC methods, a small development fee for the contract creation for most of the tasks; after a certain number of SCs, enough configurations and results will become accessible to everyone, and only custom, unprocessed circuits will require the development fee. Optionally, a dedicated blockchain network (similar to one of the existing test ones) can also be designed, with contracts that have access to real quantum hardware; its owners can decide (if necessary) the value of the virtual coin in connection to a real-world currency. For our experiments, we selected the Solidity language for the development of SCs, and Python for the development and simulation of quantum circuits, with the help of the Qiskit framework, an open-source library for quantum processing developed by IBM.

Keywords: quantum computing, blockchain, smart contract, Solidity, decentralized app.

1. INTRODUCTION

Two of the most innovative research areas that have emerged over the last couple of years are the quantum research area, especially thanks to recent developments of quantum processors, and blockchain.

We can view blockchain as a way of storing and interacting with data in a decentralized and secure manner, using a public ledger (see [1] for details).

The advantage, in terms of security, comes from the fact that data are split into multiple blocks, which, after being added to the blockchain, can no longer be changed. Blockchain does not come down just to virtual currency (cryptocurrency) as means of potential investment, as massively advertised – it can be used in the context of decentralized trust, as there is no main entity that other parties need to interact with; it can also be used as a voting mechanism. The consensus algorithms of a blockchain network guarantee that all participants adhere to the decided rules. There are multiple classifications for blockchain types, but they can generally be grouped into three categories (as presented in [1]): public blockchains – there are no limitations for users that want to participate, private blockchain – the participants are selected by the leading organization, and federated or consortium blockchains – a given number of nodes are in charge of the consensus process.

Cyberattacks and blockchain security in general are other concepts that need to be considered when dealing with blockchain networks. Hackers have managed to manipulate a certain number of vulnerabilities (see [2] for details), and some of these methods are presented as follows: one of them is represented by phishing attacks – these are attempts to trick users into giving their credentials. In routing attacks, the hackers intercept the data transfers from sent blocks; sybil attacks take advantage of the idea to flood the system, with hackers developing false identities, and in 51% attacks, if a large group of users/miners control more than half the system, then they are able to manipulate the ledger.

2. RELATED WORKS

When it comes to advances in the blockchain research area, we mention paper [3], which shows the general concepts and analyzes the applied fields and research themes of blockchain; it also presents an overview of the three generations of blockchain, while also offering an insight into the potential future research areas of blockchain technology and benefits, seen from a business point of view. For the quantum resistance in blockchain networks, paper [4] analyzes the impact that quantum technology will have on existing cryptographic algorithms. It is focused more on post-quantum protocols and techniques, post-quantum blockchain networks and the migration towards a quantum-resistant cryptographic universe (with post-quantum certificates, signing using post-quantum keys and verification of post-quantum signatures).

There are also papers that present various approaches to the world of Internet of Things (IoT); in paper [5], the authors present the security of IoT solutions from different research papers, and propose a classification based on the type of solution: blockchain, machine learning, cryptography and quantum computing. For blockchain solutions, the authors consider multiple parameters: the technique (such as Proof of Work, or a certain cryptographic algorithm), the application area (industrial, cloud-based), the main goal, the

theoretical or software frameworks used to validate the solution, the types of metrics, along with the advantages and disadvantages. A quantum blockchain-driven framework for Web 3.0 is described in paper [6]; the authors present the quantum cryptographic algorithms for Web 3.0: the quantum identity authentication (QIA), quantum consensus mechanisms, quantum block verification and propagation; they also discuss the infrastructure, and the quantum-blockchain services. The underlying concepts are based on quantum communication (quantum key distribution (QKD)), quantum random number generators (QRNGs), and so on; the applications can be grouped in smart cities, healthcare and Metaverse.

Developments have also been proposed in the healthcare system. Paper [7] proposes a blockchain security model, with a new framework named Consultative Transaction Key Generation and Management (CTKGM), which further develops the Quantum Trust Reconciliation Agreement Model (QTRAM); the idea is to establish a secure line of transmission between patients and the medical staff (the general healthcare center). A general overview of blockchain in healthcare can be found in paper [8], where the authors, after presenting multiple classifications for the existing research papers, discuss the use cases of blockchain, such as management of electronic medical records (EMRs), drug supply chains, remote patient monitoring (RPM), and others. They also take into account the limitations regarding data security, scalability and interaction with the patients; the immutability of blockchain can also pose some problems with the current legislation.

There are also research concepts that try to explore methods of building quantum circuits that can be further utilized in a quantum blockchain environment. For example, paper [9] analyzes the idea of two parties that want to exchange quantum goods, a transfer that is also validated by a randomly chosen third entity. It describes a circuit for this purpose, where 16 qubits are required, in order to describe the type of good selected by each entity, its quantity, together with validation and ancilla qubits; an alternative circuit is also studied, to take better advantage of the quantum properties.

This proposed paper analyzes the quantum information processing research area from a hybrid perspective; the underlying concepts still require a connection to a quantum computer (or simulator), but it remembers the connection details to these servers using smart contracts. They allow us to build custom quantum circuits, and use a connection that is public, secure (with the help of quantum oracles), and stored on the blockchain.

3. ORIGINALITY

The objective of this project is to present the details for the design of a system which allows users to extract quantum information, using a blockchain network. By quantum information we refer to various parameters, such as quantum circuit, probability histogram, number of qubits, number of gates, simulation time, error, and so on. With the help of

SCs, this process can achieve superior characteristics when compared to a classical solution, as it makes the code visibility of the SCs public, allows users to interact with them both in a simple (visual) manner and from a programmer's point of view, shares the quantum results with the community, and stores them for further development and analysis; such an approach helps promote the understanding of the concepts, their possible implementation, and usage of quantum technologies.

The main use case is described as follows: user A decides to interact with our proposed system, with the purpose of obtaining the quantum circuit and simulation results for a particular algorithm (it can be a well-known protocol or a new idea, without any analyzed circuit behind it). The first step is a custom search request, sent by the user to a special program that works with the SCs recorded in our system (for processing quantum data), to which we will refer to as the Smart Contract Manager (SCM); this request is sent either from the command line, or by using a web interface for the SCM. Once received, the manager finds a total or partial match using the data from the request, by searching in a quantum smart contract database (QCDB) – a database that contains information which links the SC addresses to their corresponding description, along with the quantum parameters. The QCDB itself is classical in its implementation, and its records can be accessed through SCM, which can also offer users an overview of the local collection that connects the existing SCs to their corresponding quantum circuits.

The search process has two possible outcomes: if a suitable contract is found, then the user receives its blockchain address – more than one can be recommended (by the SCM), depending on the parameters selected by the user; for example, a researcher that might want to evaluate certain performance metrics could decide to filter the circuits with a specific number of qubits, which is likely to yield multiple results.

If no potential matches are found, then a special request for a SC creation is added (posted) to the future development collection. A specialized developer from the community then views and applies to solve the request, for a certain fee (paid in the blockchain currency), which can be either specified by the user and part of the request, or automatically generated by the SCM. Once selected, the developer is given a deadline to finish the project, and upload the SC to the dedicated blockchain, together with the source code; a new entry, containing the description of this SC is also added to the QCDB. The SCM is also aware of the QCDB update, and issues a notification to the initial user, who is now able to interact with this new SC (call its public functions, with the specified parameters).

Another feature that is necessary, and allows us to be fair, by granting different views for the overall SCs development, is a score system; if multiple developers submit their intentions, the SCM chooses the winning developer, based on his/her reputation. Numerous approaches can be used to enforce the fact that the developers do not take advantage of the system; a trust score for each developer guarantees that if the development of the SC is not

finished in time, he/she will be penalized (a decrease of the score); otherwise, he will be incentivized (an increase of the score).

This described configuration is presented in Figure 1.

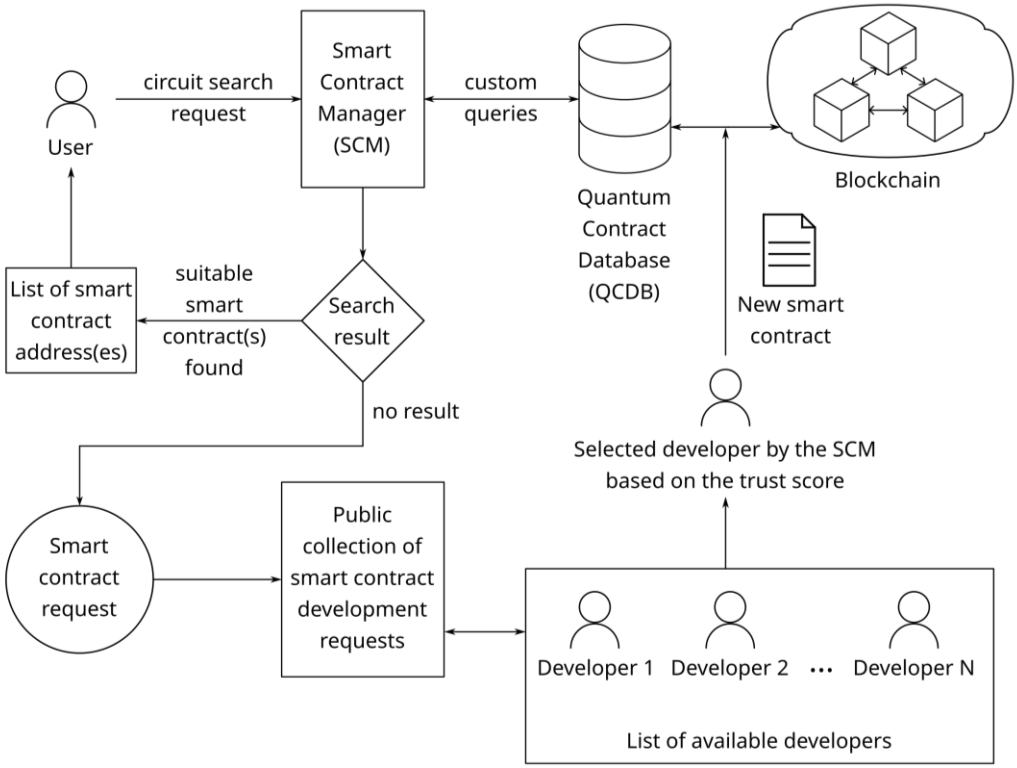

Figure 1. The general diagram of the proposed concepts presents the key elements such as the Smart Contract Manager (SCM) and the Quantum Contract Database (QCDB).

The role of the SCM is summarized as follows: first, it acts as an intermediary between the user and all the other entities that are part of the system (the QCDB, the blockchain and the developers). The SCM allows users to execute requests, which are translated to queries to the QCDB, such as extracting (selecting) certain SCs and filtering the desired ones by different criteria. If the search fields match at least one entry of the database, then the SCM returns a list of smart contract addresses (directly or parsed using a JSON file). Otherwise, if the search returns no results, it adds a new public request for the development of a SC. The SCM allows, after a time frame, to evaluate the applying developers, and then to select one automatically based on the trust score (and availability, depending on the contracts that he/she has already been selected for). Once finished, the SCM updates the QCDB with the SC address and the appropriate quantum parameters. The SCM also overlooks/verifies the upload of the SC to the blockchain. After the final step, it also notifies the user of the new SC that is now available (and its address); the SCM also updates the trust score of the publishing developer (as it stores a list of known developers and their score).

There are two main ideas that describe how the user can interact with the SC:

1. In the ideal case, we want the parameters that the user configures to be part of a special/custom request that describes a quantum circuit. The APIs for these systems vary depending on the company that shares access to their quantum devices (or simulators), so no universal formula can be proposed. This could be achieved by using a special blockchain oracle (see [10]), which allows us to gather data in a safe and decentralized manner (such as Chainlink [11] and Provable [12]). In our particular case, the goal is to use the oracle to forward the user request to IBM Quantum Systems – here, the quantum circuit is simulated, and the simulation results are returned. This approach carries a potential security risk, as different actors could use such a SC as a Distributed Denial of Service (DDoS) attack; this concern should be mitigated at the end system or at the oracle layer, together with the general security measures against such attacks.

2. As a backup solution, if, due to various problems, the blockchain oracle cannot be accessed, or presents difficulties in gathering data from the quantum system, then the SC should return the actual processed parameters, which the user should then manually input on the quantum computer's terminal (or by using its development framework). For example, if the user desired to implement a quantum algorithm on a selected number of qubits, then the SC could indicate the sequence of quantum gates that needs to be applied to obtain the corresponding circuit. This is less efficient and should only be used as a last resort.

In this paper, we discuss the first solution, as it automates the process, by directly connecting the SCs to the real quantum system. Figure 2 schematically presents this process.

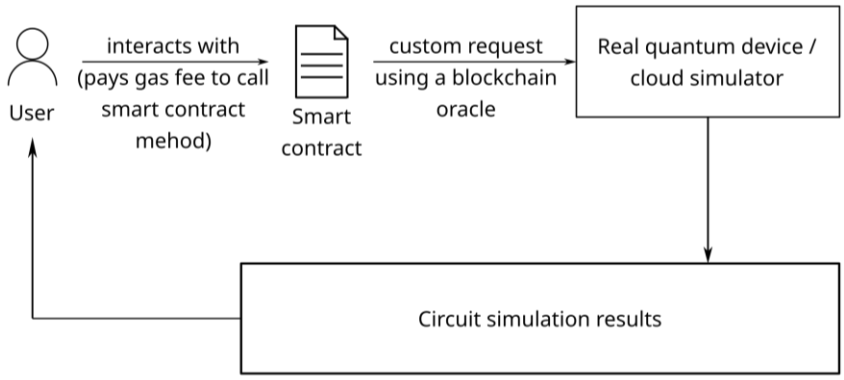

Figure 2. A diagram that presents the interaction between the user and the quantum processing layer; this is achieved using smart contracts.

The details of the user request for the SC development (that can be posted on a dedicated website) are non-standardized. Although it is possible to add some default fields (such as number of qubits, selected quantum gates and so on), the actual description of the problem is problem-specific; an

experienced quantum researcher or professor and a SC developer need to understand the actual requirements, their context, and how the contract should be structured in order to implement the selected protocol, parameterized, if possible.

Please note that all concepts regarding gas payment and fees are not designed with the purpose of gathering Ethereum whatsoever but serve to reflect the blockchain principles. The idea is to use a researcher-acknowledged currency (or a test one), valid in academia projects, and on researcher blockchain networks, with the hope of discovering talents and encouraging quantum development; concepts that stand behind this paper aim at creating an ecosystem of users that value knowledge, hard-work and resource sharing, especially in terms of quantum technology and concept awareness.

We would like to emphasize another key factor which adds flexibility to our design: the actual quantum computing part is done outside the proposed blockchain. This presents several advantages – if any change occurs on any used quantum processor, the connection itself should still work without any problems; the users are also not required to know the details of the quantum devices, as the smart contract manager oversees the connection to the quantum platform.

4. SYSTEM DESIGN

This section presents the selected software tools, the ideas behind the database concepts, and the trust mechanism; after this, we show the steps required to develop a system similar to the one proposed in this paper.

4.1 Software tools

In order to accelerate the development process, the SQLite library was selected for the implementation of the QCDB, as its properties recommend it as one of the easiest to configure databases in the world (see the SQLite official website, [13]). The programming language selected for the project is Python, offering support for a large number of libraries, that facilitate the general research process. Details about connecting to a sqlite3 database file using Python and then interacting with its records can be found on the official documentation page [14].

We also mention the support that IBM offers researchers in designing and testing quantum circuits, by allowing their simulation on real quantum processors in the cloud, on special simulators, or on the local machine. The probability histogram obtained after simulating each circuit is of great help, as it validates the proposed concepts and allows for adequate modifications; for details on simulating these circuits on IBM Quantum in the browser, or for the documentation of the open-source SDK called Qiskit, written in Python, please consult [15] and [16] (the official Qiskit GitHub page).

The SCs are written in Solidity (see [17] and [18]), implemented with the help of Foundry, a smart contract development toolchain (see [19] and

[20]); a dashboard for a test network, used for development purposes, can be accessed with the Alchemy framework, AlchemySDK (see [21]).

4.2 Database (QCDB) design

There are multiple ways to imagine the fields that can be saved in a local database for easier browsing and filtering of the designed SCs. One such database design is presented below, containing three tables: *Developer*, *Quantum_circuit* and *Smart_contract*. Each of them has the following attributes:

1. *Developer*:

• id <number> – primary key;

• dev_address <string> – blockchain address of software developer;

• dev_score <number> – a rating (number) that indicates the trust score of the current developer, based on his/her activity, and user reviews;

• dev_info <string> – information about the developer (age, coding background, rating, and so on; this field can be further broken down into more fields, if developer filtering becomes essential);

2. *Smart_contract*:

• id <number> – primary key;

• sc_address <string> – SC blockchain address;

• sc_info <string> – additional information about the SC or notes added by the developer, description and limitations;

• dev_id <number> – foreign key, id from the Developer table;

3. *Quantum_circuit*:

• id <number> – primary key;

• sc_id <number> – foreign key, id from the Smart_contract table;

• nr_qubits <int> – number of qubits used in the circuit;

• nr_gates <int> – number of quantum gates used in the circuit;

• qc_info <string> – information about the quantum circuit, description and use-cases.

The *Quantum_circuit* table should be open for improvement and development, as parameters that become essential for the search process, as well as computing various metrics, will be required by researchers over time (such as processing time, gate error, quantum volume, selection of gates). On a different note, the actual code behind the SC (the .sol file) can be saved directly, using a platform similar to the existing blockchain explorers (see [22] for example, a Sepolia Blockchain Explorer).

4.3. Trust system

Different systems can be proposed for calculating the scores of developers, each with its own advantages and disadvantages. We present a simple system that modifies the trust score (or rating) for each developer after posting a SC.

For example, each user can offer a rating to their requested contracts, with the following available options: [*good*, *decent*, *bad*]. The system can also

validate whether the deadline was respected or not. Depending on (at least) these factors, a local review score can be calculated and added to the general score of the developer. A local "decay" factor, that slowly reduces the score of inactive developers over time, can also be suggested; more improvements, such as review granularity, can be implemented as the system evolves. We also consider a low and high ceiling for each developer's score, with a starting score for new developers of 0. Table 1 shows a simple local score evaluation, after a SC was proposed, and then reviewed by the end user who requested its design.

Table 1. Evaluation score of the current smart contract development

Deadline status \ User review	**Deadline not respected**	**Deadline respected**
bad	-10	-10
decent	-5	+5
good	-5	+10

4.4. Development workflow

What still needs to be clarified is how we can go from the quantum circuit to calling the corresponding SC methods that retrieve the simulation results. The workflow requires putting together all the ideas, from the actual description of the quantum circuits to the user calls to these contracts (with parameters, where available); this process can be defined with the following steps:

1. The first step is to design the Python code for the desired circuit, with the associated request to the targeted quantum system (using the Qiskit API). The goal is to obtain an identifier (URL) that points to the API for the simulation of the circuit.

2. We use the Solidity language to create the SC, which basically integrates the custom request (to simulate the quantum circuit) sent to the external platform into a certain SC; we then upload it to a test network (locally, or by using the Sepolia test network, for example).

3. Users can now call the simulation method, with the parameters decided by the developer, which in turn, causes the oracle to interact with the API from the quantum service provider. After a certain amount of time, depending on the load of the quantum device, the call result is returned (most likely in the shape of data serialization formats, such as JSON or XML).

As mentioned before, the oracle component of the solution guarantees that data are gathered in a decentralized manner.

5. EXPERIMENT AND ANALYSIS

This section describes the steps that were taken for a custom SC, used to generate a simple entanglement circuit on two qubits.

5.1 Explanations on the API from IBM Cloud

The proposed plan cannot be implemented in a straightforward manner with the Qiskit library if a local simulator is targeted, because the Solidity language does not have access to external Python frameworks (and the blockchain does not directly allow its SCs to access external data by design, hence the need for quantum oracles). However, we found a solution after investigating the general ways of remotely accessing IBM's devices (real and quantum simulators), presented in detail in the documentation regarding the configuration of the connection (see [23]). The two IBM Quantum channels are IBM Quantum Platform and IBM Cloud.

They both rely on using the Qiskit Runtime Client. IBM Cloud allows users to access various services remotely, and the one of interest to the quantum research area is Qiskit Runtime, that allows users to create custom API calls using the IBM Quantum Qiskit Runtime API; it is in beta phase at the time of writing (see [24] for details regarding the available commands). The users can interact with the jobs, backends, sessions and instances (and more). The Jobs section is of critical importance (to the SC developer), as it allows us to show the jobs, run, cancel, delete a job, list the results, and other options.

The SC developer, after understanding and designing the quantum circuit, can configure the equivalent API call and insert it in a SC, so that when a user interacts with one of its methods, the request is redirected to the IBM systems. Depending on the circuit, the user should provide multiple parameters, but at the very least, two connection parameters are necessary: the Service Cloud Resource Name (CRN), and an IBM Cloud API key (or an IBM Cloud Identity and Access Management/IAM token).

A development idea (for companies working with quantum technologies) is perhaps a public release of a workstation that can be accessed through a single common token, for research purposes (which would be recommended in the future with the hope of better integrating the connection between one or more quantum systems and SCs). This suggestion is given in the context of keeping information secure, as providing tokens as parameters is not a best practice and should only be done with test accounts or in a testing environment (also notice that embedding certain tokens in a SC is not safe, nor efficient). For the rest of the paper, we assume, for research purposes, that the CRN and API key can be safely shared with the developer (and with the quantum community, if need be).

5.2. Design of the SC for the entanglement circuit

As previously mentioned, the developer needs to have access to certain data from the user (the CRN and IBM API token) – they can be provided on a dedicated platform. The developer can approach his/her task in (at least) two ways; if he only wants to use the API, then he needs to configure the parameters header section (*params*), which might prove unnecessarily tricky. The easier way is to use the Python programming language and first code the

quantum circuit in Qiskit, by using the *QiskitRuntimeService* class with the IBM Cloud option, using the provided CRN and API token data. Then, after configuring the circuit, he can send it to the cloud for simulation, obtaining the job id in the process. After this step, he can easily retrieve all the jobs (if more than one test was requested from that user) and filter them by the job id.

For a system of two qubits with a classical entanglement setup, using a Hadamard and a CNOT gate, the circuit can be consulted in Figure 3.

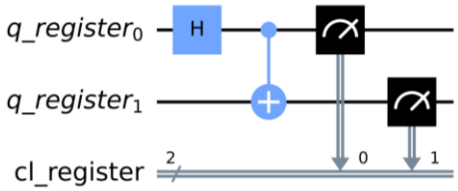

Figure 3. A simple entanglement circuit designed using Qiskit – it requires two qubits, a Hadamard and a CNOT gate.

After configuring *QiksitRuntimeService* in Python, and the specific circuit configuration, a job can be added in the execution queue of the desired device in the Cloud. By using a tool such as curl (*cURL*) to interact with data from URLs (see [25] for details), the developer can write the command line required to retrieve the last 20 jobs in the following way:

```
curl -g --request GET "https://us-east.quantum-computing.cloud.ibm.com/jobs?limit=20&offset=-22996455&pending=false" --header "Service-CRN: test_crn" --header "Authorization: apikey test_key"
```

(for privacy reasons, the actual CRN and API key were replaced in this command with *test_crn* and *test_key*).

The developer can then obtain all the available information about the job (details, results, logs, metrics), but the component containing the results (the distribution of probabilities) is the most important. Notice the job id, *clpn0nbnj01vtk4mho70*, in the URL; this id is obtained after running the code (submitting the job) in Python, and it is also available in the results obtained after running previous curl command. This can be done as follows:

```
curl -g --request GET "https://us-east.quantum-computing.cloud.ibm.com/jobs/clpn0nbnj01vtk4mho70/results" --header "Service-CRN: test_crn" --header "Authorization: apikey test_key"
```

In this case, the simulation took place on the simulator called *ibmq_qasm_simulator* (with 32 qubits), using 4000 program executions (called shots). The format of the results is a JSON file, containing two main tags: *metadata* and *quasi_dists*. The *quasi_dists* tag refers to a dictionary

containing all the possible quantum results (here, ideally, only the collapsed 00 and 11 states, and their probabilities):

"quasi_dists": [{"00": 0.4885, "11": 0.5115}].

We would like to mention that there are other Python packages available that can manage GET requests (as alternatives to the *curl* command). The representation of the probabilities from the obtained *quasi_dists* dictionary using the API call is shown in Figure 4.

The actual content of the SC comes down to two parts. The retrieval of the job results is mandatory; the second one, that can be optionally called first, would involve running the job again or creating another similar one (this idea is not relevant in our paper, but it could be useful in different scenarios).

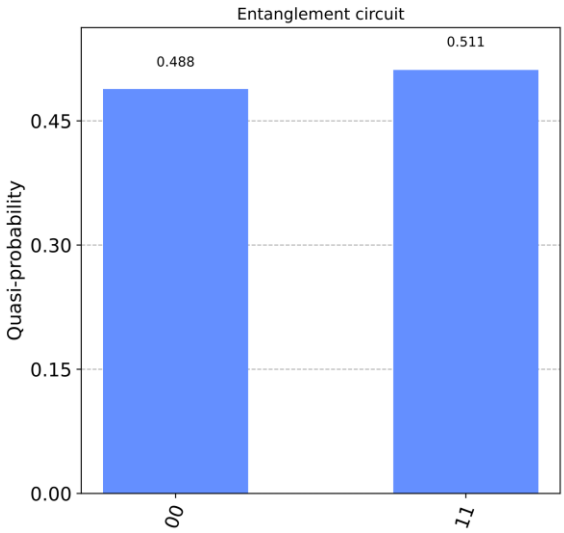

Figure 4. The measurement results for the entanglement circuit (on ibmq_qasm_simulator) – both obtained states are close to the theoretical probability of 0.5.

The process of testing the retrieval of a custom JSON response generated by a URL request (including the one obtained using the API from IBM Cloud) can be achieved with the help of the Foundry-Chainlink Toolkit (also in beta phase at the time of writing, see [26]). As its documentation presents, it allows us to test our SC ideas on a local Chainlink node, built using the Foundry framework. An RPC node can be configured using the Anvil component from Foundry (the *make fct-anvil* command), and we can simulate the testing environment, a Chainlink cluster, composed by default of 5 Chainlink nodes (by running the *make fct-init* command); these nodes can be identified in the Docker environment, and appear as *foundry-chainlink-node1* up to *foundry-chainlink-node5*.

In terms of pseudocode, the general ideas for a SC are presented in Figure 5.

```
check = verify_authorization_credentials(encrypted_calldata, …)

if check is true then
    local_quantum_circuit_address is initialized from a SC parameter

    // The circuit is basically bound to the SC address,
    // but can optionally present additional parameters.
    // The actual search that finds the corresponding SC address
    // is done outside the SC code (using the QCDB).

    if local_quantum_circuit_address is valid then
        results = call_to_IBM(local_quantum_circuit_address, …)
        return results
    end
end
```

Figure 5. The pseudocode which describes the main ideas behind the Smart Contract.

5.3. Comparison to a classical simulation workflow

By considering the mentioned aspects, we can summarize the main advantages and drawbacks of a distributed (blockchain) solution for the analysis of quantum algorithms, with their respective circuits (and potential storage of the results), when compared to a classical one. These ideas are presented in Table 2.

Table 2. Comparison between the proposed solution and a classical approach

Criteria	Blockchain solution	Classical approach
Research and development	Low – for most algorithms, as corresponding SCs will already be available, no programming language is needed. It only requires the command-line or a custom interface to access the needed circuits (it integrates the discussed concepts as a library) for the already analyzed algorithms. For a new algorithm, a developer needs to accept the task, program the circuit, and write a SC that accesses its simulation results, and then upload it to the blockchain.	Medium-high – the user needs to have programming knowledge, and code his/her own circuits for an in-depth analysis (or use an existing drag & drop interface). This also requires the installation and configuration of a quantum development framework.
User cost	Low-none – transaction fees (gas) are needed to call the functions using a development coin.	Low-none – no cost or costs required by the quantum cloud system (for some of the systems, with basic functionality).
Infrastructure cost	Low-none – the already existing testing (or real) blockchain networks can be used to store the SCs that call the quantum systems.	No additional costs.

Criteria	Blockchain solution	Classical approach
Solution complexity	Low-medium – a new complexity layer is added, which requires using the methods from the SCs.	Low – by directly accessing the existing solutions.
Time	Low – the time delay induced by using a SC is usually insignificant (although dependent on the network), and the actual simulation time is not affected.	Low – just the quantum simulation time.
Security	High – by using the blockchain approach, the architecture offers an inherently higher level of security, together with the application of oracles.	Low – if an API is used, it is difficult to validate the origin of the parsed results (the data could be intercepted and then altered).
Legacy	High – the proposed design stores the SCs used to analyze certain classes of quantum algorithms in the blockchain; this allows all researchers to quickly access the desired simulations, by calling the SC methods, which in turn create a request to the quantum systems. The local database and the automated request speed up the research process.	Low-none – unless the user saves the circuit and the simulation results, and then shares them with other researchers, the simulation results remain private; it is difficult to optimize quantum knowledge sharing with the community, without the proper infrastructure.

It can be concluded that the most impact would be found in the following criteria: *Research and development* (with a lower barrier of entry to hands-on algorithms and devices at the same place), *Security* (by using the blockchain networks and oracles) and *Legacy* (as a certified, better way, to store details about circuit configurations).

We would like to mention the fact that although this infrastructure is presented as a layer that operates on top of the existing classical blockchain mechanisms, a different approach can also be considered. The underlying system for our SCs does not necessarily need to be classical – we could propose quantum alternatives as parts of our design, that have been previously analyzed by researchers in the area. As a core validation component, we could rely on a combination between asymmetric encryption and a stake vote consensus algorithm; this is presented in paper [27], where the authors show the quantum circuits for their proposed operations, describing the quantum circuits to sign a transaction (using a quantum one-way function), the verification process, and presenting the security analysis

of their scheme. Another alternative to the classical blockchain would be to design a quantum blockchain using the entanglement in time as the core property for implementing the chain concept (see paper [28] for details), where its authors expand the concept of GHz states, building on the time stamps for each block.

In terms of potential future development of this project (with the concepts described here), we can envision two directions: the first one represents an investigation of multiple quantum frameworks, and perhaps the creation of a unified (or higher-level) API, that can allow users to select the actual platform on which they want to run their quantum circuits. Another idea is based on the research of multiple blockchain technologies; their selection inevitably impacts the performance of the proposed design.

6. CONCLUSION

The presented paper describes a method of combining the tools offered by various platforms, such as IBM Cloud for quantum computing, Foundry for blockchain development, Chainlink for quantum oracles, together with the Python programming language that interacts with Qiksit; information about the circuits and developers is stored in a database (on a classical server) for easier browsing, which can be updated with the description of quantum circuits when deploying each smart contract to the blockchain. All these tools lay the foundation of a new way to create quantum circuits and test them on real quantum devices or simulators. We can then retrieve the results by using the IBM Cloud API and store the access tokens to the circuit and their results in public fashion with the help of smart contracts on the blockchain.

Using the presented steps, multiple quantum algorithms (known or custom) can be developed, and a collection of algorithms with their actual simulation results can be publicly obtained. Further developing such a framework could help us grow the emerging field of applied quantum computing and offer a better overview of quantum protocols (circuit model) by giving users theoretical information, together with hands-on simulation results.

REFERENCES

[1] **What is Blockchain? | Oracle**, *Oracle (Oracle Cloud Infrastructure, OCI)*, [Online]. Available: https://www.oracle.com/uk/blockchain/what-is-blockchain/.

[2] **What is Blockchain Security? | IBM**, *IBM*, [Online]. Available: https://www.ibm.com/topics/blockchain-security.

[3] M. Xu, X. Chen and G. Kou, **A systematic review of blockchain**, *Financial Innov.*, vol. 5, no. 27, pp. 1-14, 2019.

[4] M. Allende, D.L. León, S. Cerón, A. Pareja, E. Pacheco, A. Leal, M. Da Silva, A. Pardo, D. Jones, D.J. Worrall, B. Merriman, J. Gilmore, N. Kitchener and

S.E. Venegas-Andraca, **Quantum-resistance in blockchain networks**, *Sci. Rep.*, vol. 13, no. 5664, pp. 1-23, 2023.

[5] S. Cherbal, A. Zier, S. Hebal, L. Louail and B. Annane, **Security in internet of things: a review on approaches based on blockchain, machine learning, cryptography, and quantum computing**, *J. Supercomput.*, vol. 80, no. 3, pp. 1-79, 2023.

[6] M. Xu, X. Ren, D. Niyato, J. Kang, C. Qiu, Z. Xiong, X. Wang and V.C.M. Leung, **When Quantum Information Technologies Meet Blockchain in Web 3.0**, *IEEE Network*, vol. 38, no. 2, pp. 255-263, 2023.

[7] Selvarajan and H. Mouratidis, **A quantum trust and consultative transaction based blockchain cybersecurity model for healthcare systems**, *Sci. Rep.,* vol. 13, no. 7107, pp. 1-21, 2023.

[8] C. C. Agbo, Q. H. Mahmoud and J. M. Eklund, **Blockchain Technology in Healthcare: A Systematic Review**, *Healthcare*, vol. 7 (2), no. 56, pp. 1-30, 2019.

[9] A.-G. Tudorache, **Design of an Exchange Protocol for the Quantum Blockchain**, *Mathematics*, vol. 10 (21), no. 3986, pp. 1-14, 2022.

[10] **What Is a Blockchain Oracle?**, *Chainlink Foundation*, [Online]. Available: https://chain.link/education/blockchain-oracles.

[11] **Chainlink: The Industry-Standard Web3 Services Platform**, *Chainlink Foundation*, [Online]. Available: https://chain.link.

[12] **Provable - the provably honest oracle service**, *Provable Things Ltd.*, [Online]. Available: https://web.archive.org/web/20231130211804/https://app.provable.xyz/home/features.

[13] **SQLite**, *Hwaci*, [Online]. Available: https://www.hwaci.com/sw/sqlite/index.html.

[14] **sqlite3 — DB-API 2.0 interface for SQLite databases**, *Python Software Foundation*, [Online]. Available: https://docs.python.org/3/library/sqlite3.html.

[15] **IBM Quantum Documentation**, *IBM*, [Online]. Available: https://docs.quantum-computing.ibm.com/.

[16] **GitHub - Qiskit/qiskit**, *IBM*, [Online]. Available: https://github.com/Qiskit/qiskit.

[17] **Solidity documentation**, *The Solidity Authors*, [Online]. Available: https://docs.soliditylang.org/en/latest/index.html.

[18] **GitHub - ethereum/solidity**, *open-source, with the core team sponsored by the Ethereum Foundation*, [Online]. Available: https://github.com/ethereum/solidity.

[19] **Introduction - Foundry Book**, *open-source library*, [Online]. Available: https://book.getfoundry.sh/.

[20] **GitHub - foundry-rs/foundry**, *Paradigm and the Rust Ethereum open-source community*, [Online]. Available: https://github.com/foundry-rs/foundry.
[21] **Alchemy - the web3 development platform**, *Alchemy Insights, Inc*, [Online]. Available: https://www.alchemy.com/.
[22] **TESTNET Sepolia (ETH) Blockchain Explorer**, *Etherscan 2025 (Sepolia)*, [Online]. Available: https://sepolia.etherscan.io/.
[23] **Select and set up an IBM Quantum channel | IBM Quantum Documentation**, *IBM*, [Online]. Available: https://docs.quantum.ibm.com/start/setup-channel.
[24] **IBM Quantum Qiskit Runtime API | IBM Cloud API Docs**, *IBM Cloud*, [Online]. Available: https://cloud.ibm.com/apidocs/quantum-computing.
[25] **curl - command line tool and library for transferring data with URLs**, *open-source software*, [Online]. Available: https://curl.se/.
[26] **GitHub - smartcontractkit/foundry-chainlink-toolkit: A plugin to spin up local Chainlink node with Foundry**, open-source, [Online]. Available: https://github.com/smartcontractkit/foundry-chainlink-toolkit.
[27] W. Wang, Y. Yu and L. Du, **Quantum blockchain based on asymmetric quantum encryption and a stake vote consensus algorithm**, *Sci. Rep.*, vol. 12, no. 8606, pp. 1-12, 2022.
[28] D. Rajan and M. Visser, **Quantum Blockchain Using Entanglement in Time**, *Quantum Rep.*, vol. 1, no. 1, pp. 3-11, 2019.