Open Access License Notice

Integration of Blockchain Concepts into Computer Science Curriculum

Eric Sakk
Department of Computer Science
Center for the Study of Blockchain and Financial Technology
Morgan State University
Baltimore, MD, USA
eric.sakk@morgan.edu

Shuangbao Paul Wang
Department of Computer Science
Morgan State University
Baltimore, MD, USA
shuangbao.wang@morgan.edu

***Abstract*—In this work, we consider the nexus between blockchain technology and computer science curriculum. While it is possible to introduce the blockchain paradigm using a single course, the depth of a single topic can often be sacrificed at the expense of covering a breadth of information. As blockchain is an emerging technology, it is important to embed various concepts throughout the undergraduate curriculum with the depth necessary to reinforce each facet. Using a just in time approach, we define exactly where and how blockchain topics relevant to computer science should be introduced. As a means for active learning pedagogy, we introduce a lab framework for students to gain hands-on experience. Finally, we describe collaborations with industry to provide mentorship and internship opportunities.**

Keywords—blockchain, hash function, proof of work, computer science introduction

I. Introduction

Blockchain is an emerging technology that is transforming the way business and financial institutions process transactions. Given this current state of affairs, it is critical that computer science training involve technical aspects of such processes. In addition to tutorials and instruction available from industry [8, 9], many academic institutions have begun to offer curricula containing specific courses dealing with blockchain technology [4, 10]. While the offering of full courses through academia or industry is sensible, the applications are so vast that it can become difficult to achieve the necessary technical depth. Within the computer science major, our approach is to weave aspects of blockchain technology throughout the existing undergraduate curriculum. This allows for concept immersion and reinforcement through the revisiting of ideas at various stages with increasing depth.

With regard to computer science training, key topics most relevant to blockchain technologies are found within courses dealing with:

- Introductory and Advanced Programming Techniques
- Data Structures
- Computer Networks
- Network Security
- Cloud Computing
- Cryptography
- Machine Learning and Artificial Intelligence

The main goal of this work is to demonstrate how blockchain concepts can be integrated into these key topics. For the purposes of illustration, examples are presented using the Python programming language.

II. Introductory Programming Techniques

At the heart of any programming curriculum are basic constructs necessary for algorithm implementation. Beginning at the introductory level, key concepts to be covered generally include:

- Variable types
- Arithmetic and logical operations
- Loops
- Functions
- Lists
- Nested lists and nested iterations
- Methods and basic object oriented programming

Instead of introducing lists in Python, other programming languages might include rudimentary data structures exhibiting similar behaviors such as arrays, structures or cell arrays.

To emphasize the practice of the above concepts, we propose the implementation of a blockchain cryptocurrency example that grows in complexity as new programming constructs are introduced and developed. Key components of an introductory level example include:

- Variable types for the blockchain participants requiring both sender information and receiver information
- Transaction information

- Account balance computations
- Verifying transactions before adding them to a set of open transactions
- Adding a new block to the blockchain with a validated set of open transactions
- Methods for hash function implementation
- Nested list data structure for storing the complete blockchain

These initial components are enough to demonstrate simple cryptocurrency transactions from which students can glean understanding of how a blockchain works. Furthermore, existing libraries for Python make for fast implementation of operations involving a fair amount of complexity such as SHA-256. For instance, the Python library '*hashlib'* [2] can be used to implement hashes and message digests with straightforward method calls that can easily be applied at the freshman level. When used in tandem with the 'json' library [3], it becomes possible to encode any data structure available in Python that one might choose to represent the blockchain. For example, assume '*encodedBCBlock*' represents a variable encoded from the most current block within the simulated blockchain. Then a command as simple as

```
hashlib.sha256(encodedBCBlock).hexdigest()
```

can be used to generate the associated block hash formatted with hexadecimal characters. This is the beauty of using Python as an introductory language. Many computationally intensive operations that would be unraveled in greater depth within upper level courses can be easily programmed using existing Python libraries. A novice programmer can then experience immediate results that do not get bogged down with the finer details of an intense computation.

Figure 1 shows an abbreviated version of a menu shell in the form of a while loop for achieving blockchain functionality. This snippet of code encompasses many of the key programming concepts outlined above. After introducing variable types, lists, boolean operations, loops, and functions, the blockchain example can be initiated. The code for each menu item can then be successively added as the details of each new operation is introduced. As a concrete example from Figure 1, choice number five allows a new block to be added to the blockchain. As part of this process, Proof of Work must be established.

Figure 2 shows some sample Python code that performs a simple iterative search for a hash containing an instance of the correct nonce. This type of code is highly instructive at the freshman level as it involves simple data structures, loops, boolean operations, and function calls applied in the context of blockchain implementation.

```
1   def input_selection():
2       usr_inputvalue  = input('Make your selection: ')
3       return usr_inputvalue
4
5   exit_the_menu_loop = False
6   while(not exit_the_menu_loop):
7       print()
8       print('Select a blockchain operation')
9       print('1: Output the complete current blockchain')
10      print('2: Output a list of current users along with their balances')
11      print('3: Verify balances for the current transaction')
12      print('4: Add a new transaction to the set of open transactions')
13      print('5: Add a new block to the blockchain')
14      print('6: Hack the blockchain')
15      print('q: Quit')
16      print()
17
18      user_vals = ['1', '2', '3', '4', '5', '6', 'q']
19      user_selection=input_selection()
20      if user_selection not in user_vals:
21          print('Invalid input, please try again')
22      elif user_selection == 'q':
23          exit_the_menu_loop = True
24
```

Fig. 1. Example 'while' loop.

```
80  def test_nonce(candidates, prev_hash, pwcounter):
81      teststr=(str(candidates)+str(prev_hash)+str(pwcounter)).encode()
82      teststr_hash=hashlib.sha256(teststr).hexdigest()
83      return teststr_hash[0:4]!='0000'
84
85  def proof_of_work(prev_block):
86      prev_hash=blockhash(prev_block)
87      pwcounter=0
88      while test_nonce(candidates, prev_hash, pwcounter):
89          pwcounter+=1
90      return pwcounter
```

Fig. 2. Proof of Work example.

III. Sophomore Level Concepts

A. Data Structures

The example from the freshman level can be expanded when discussing concepts from a typical data structures course in a number of ways. For instance, the nested list data structure could be implemented in the form a linked list where each node would correspond to a given block. In addition, algorithmic complexity examples involving various loops and methods would be a natural consequence of revisiting techniques previously introduced in earlier courses.

B. Advanced Programming Techniques

An advanced programming course should involve deeper aspects of object oriented programming (OOP) and the construction of dedicated classes. In addition to rephrasing the introductory blockchain example with more sophisticated OOP concepts, projects for this level involve the programming of smart contracts for digital property transactions. Software engineering projects naturally include the design of user interfaces for smart contract transactions similar to those available for Ethereum using, for example, Solidity [7].

IV. Upper Level Concepts

A. Computer Networks

Basic understanding of computer networks involves introducing concepts such as network protocols, network architecture, fault tolerance, and software defined networks. Hence, deeper immersion into blockchain and cryptocurrency concepts such as proof-of-work, proof-of-stake and 51% attacks naturally arise in this setting. The example introduced at the freshman and sophomore levels can now be phrased in a context where multiple network nodes have stored copies of the current blockchain. Course projects involving the mining of new blocks and analyzing various tradeoffs for achieving scalability, throughput maximization and latency minimization would be appropriate at this level.

B. Network Security, Cryptography, and Cloud Computing

Assuming computer networks as co- or prerequisite material, detailed understanding of hash functions and nonces for block mining, public key cryptography for wallets, as well as authentication and key distribution are necessary to provide the necessary depth of understanding for blockchain implementation. After mining, each block will be submitted as a new transaction. In the cloud computing course, techniques to setup AWS EC2, VPC and create interfaces are introduced where the blockchains are hosted in a cloud environment. With this material as a foundation, undergraduate research projects can be offered in order to simulate and analyze various facets of mining and blockchain security. Advanced topics such as IoT security and quantum cryptography are also introduced at this level [11, 14].

C. Deep Learning and Artificial Intelligence

This is an extremely fertile area of research [1, 5, 6], the basics of which can be discussed after basic deep learning concepts have been introduced. For instance, deep learning applications where the training data cannot be exposed (e.g. health informatics) require privacy - preserving methodologies. Blockchain architectures have recently been proposed addressing such classes of problems involving trusted data. At this level, it would be expected that students have a command of blockchain technology and would engage in simulation, projects and research involving deep learning applications of blockchain.

V. Experiential Learning via the AI and Cloud Computing Lab

Learning is an integration of interaction. The interaction might exist between learners and instructors/computers or between learners and the real world [15]. While traditional approaches tend to focus more on lectures, emerging fields in Computer Science require students to be equipped, not only with knowledge, but also skills [12, 13].

For this study, we have setup an AI, IoT, and Cloud Computing Lab for upper level students in order to teach AI,

Cloud Computing, and Cybersecurity through hands-on, experiential learning. Currently we have acquired a number of AWS DeepLens, DeepRacer, and IoT Buttons, as well as different types of Echos. Figure 3 shows the architecture of the AI lab in the AWS Cloud. It demonstrates how to process blocks submitted to the cloud with the help of AI.

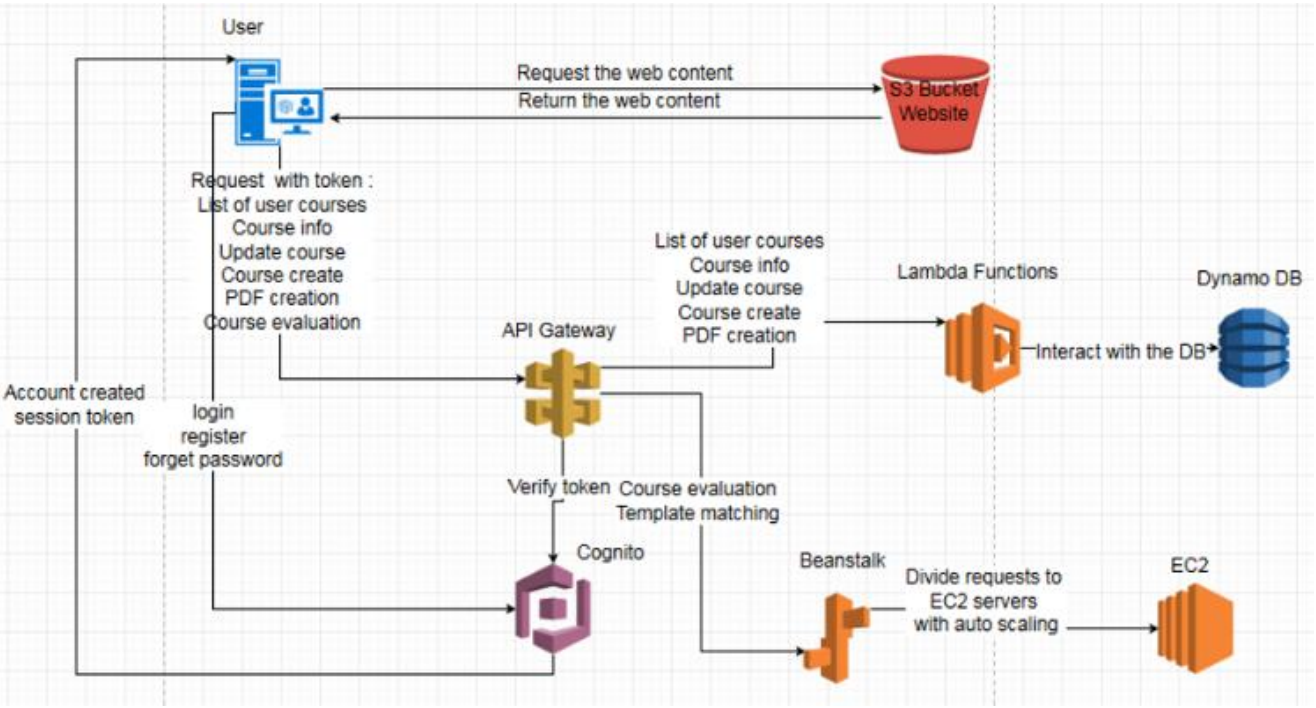

Fig. 3. Diagram of an AI-assisted Block Processing Lab in the AWS Cloud.

We are currently collaborating with Google, Facebook, and JP Morgan Chase where engineers from these companies are invited to participate in the teaching of classes, to aid in the development and mentoring of projects, and to provide summer camp and internship opportunities. A pipeline has therefore been established to study blockchain and other emerging areas in Computer Science allowing for internships and full-time employment within various high-profile tech companies.

VI. Experiments and Further Discussion

The methodology presented in this work is new to our department and a phased-in blockchain integration plan is currently under evaluation. Some courses will begin phase-in during the 2020-2021 academic year with the goal of generating academic performance data to measure student learning as a function of introducing blockchain concepts. The goal for Fall 2020 will be to compare the performance of two cohorts: one with blockchain concepts introduced and one without.

At present, we have generated data from a sample of ten students in a junior-level course entitled Introduction to Cryptography. Specifically, a course unit dealing with hash functions has been taught with blockchain as the foundational example. Furthermore, this unit allows students to numerically experiment with Python code involving proof of work. The intent is to compare student performance in this cohort against a Fall 2020 cohort. This group will not apply blockchain as the primary vehicle for introducing and applying hash functions. Anecdotal data from previous iterations of this course without blockchain suggests that the current cohort has been much more engaged when presented with the blockchain application.

With regard to best practices, various approaches have been cited involving curricular a la carte itineraries as well as global subjects adapted to each person [16]. In addition, while blockchain has been introduced into curricula involving business applications, for example [17], the distributed approach presented here of introducing blockchain into computer science curricula, to the best of our knowledge, has not been tested. We are effectively on new ground and it will take some time to infuse our courses with this new material.

VII. Conclusions

In this work, we have outlined a framework for integrating blockchain technology into all levels of a computer science curriculum. This approach addresses the 'breadth-vs-depth' dilemma that might be encountered within a single course addressing blockchain applications. In this

way, the necessary depth can be achieved over time by introducing blockchain concepts in courses where they are most appropriate. Furthermore, an AI, IoT, and Cloud Computing Lab has been established so that students can learn by doing. The main benefit achieved is a deeper understanding through practical implementation and simulation.

ACKNOWLEDGEMENTS

Funding for this work was provided in part by the Center for the Study of Blockchain and Financial Technology at Morgan State University.

REFERENCES

[1] X. Chen, C. Ji, J. Luo, W. Liao, and P. Pan Li. 2018. When Machine Learning Meets Blockchain: A Decentralized, Privacy-preserving and Secure Design. *IEEE International Conference on Big Data* (2018), 1178–1187.

[2] Python Documentation. 2020. *hashlib — Secure hashes and message digests.* https://docs.python.org/3/library/hashlib.html [Accessed Feb. 29, 2020]

[3] Python Documentation. 2020. json — *JSON encoder and decoder*. https://docs.python.org/3/library/json.html [Accessed Feb. 29, 2020]

[4] MIT. 2020. *blockchain.mit.edu*. blockchain.mit.edu/ [Accessed Feb. 29, 2020]

[5] P. Mohassel and Y. Zhan. 2017. Secureml: A system for scalable privacy-preserving machine learning. *IEEE Symposium on Security and Privacy* (2017), 19–38.

[6] R. Shokri and V. Shmatikov. 2015. Privacy-preserving deep learning. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (2015), 1310–1321.

[7] Solidity. 2020. *Solidity 0.6.1 documentation*. https://solidity.readthedocs.io/en/v0.6.1/ [Accessed Feb. 29, 2020]

[8] Global Software Support. 2020. Blockchain Explained Step by Step. https://www.globalsoftwaresupport.com/blockchain-explained-step-by-step/ [Accessed Feb. 29, 2020]

[9] Global Software Support. 2020. *Cryptocurrency (Bitcoin) Explained Step by Step*. https://www.globalsoftwaresupport.com/cryptocurrency-bitcoinexplained -step-by-step/ [Accessed Feb. 29, 2020]

[10] Stanford University. 2020. *The Stanford Center for Blockchain Research*. https://cbr.stanford.edu/ [Accessed Feb. 29, 2020]

[11] Paul Wang, A. Ali, U. Guin, and A. Skjellum. 2018. IoTCP: A Novel Trusted Computing Protocol for IoT. *The Colloquium of Information Systems Security Education (CISSE)* (2018), 165–180. Issue 1.

[12] Shuangbao Wang. 2016. Dual-Data Defense in Depth Improves SCADA Security. Signal (2016), 42–44. Issue 10.

[13] Shuangbao Wang and William Kelly. 2017. Smart Cities Architecture and Security in Cybersecurity Education. *The Colloquium of Information Systems Security Education (CISSE)* (2017). Issue 2.

[14] Shuangbao Wang, M. Rhhde, and A. Ali. 2020. Quantum Cryptography and Simulation: Tools and Techniques. *ACM Proc. of International Conference of Cryptography, Security and Privacy (ICCSP)*, pp 36-41.

[15] Shuangbao Paul Wang and William Kelly. 2017. Video-based Big Data Analytics in Cyberlearning. *Journal of Learning Analytics* 4, 2 (2017), 36–463.

[16] A.R. Bartolomé. 2020. Blockchain in Educational Methodologies. In: Burgos D. (eds) Radical Solutions and eLearning. Lecture Notes in Educational Technology. Springer, Singapore

[17] W. Dettling. 2018. How to Teach Blockchain in a Business School. In: Dornberger R. (eds) Business Information Systems and Technology 4.0. Studies in Systems, Decision and Control, vol 141. Springer, Cham