A-Noncing the Performance of Hashing Algorithms

Abstract:

Hashing concepts are an important, and often difficult, part of teaching computer science. Secure hash algorithms which are used to verify that data has not been altered via man-in-the-middle threats; it is also utilized for password protection, digital signatures, and to verify currency transactions in distributed blockchain ledgers. It is important that students receive a solid foundation in the applications of hashing algorithms. This paper presents three exercises that can be used to illustrate secure hashing concepts using three online exercises that were developed in PHP and are available for immediate use.

Secure Hashing Applications

Hashing concepts are an important, and sometimes difficult part of teaching computer science. The family of secure hash algorithms (SHA) used in cryptography have a variety of overlapping uses including:

- **Ensures that data on servers has not been changed**. Hashing significantly reduces processing time. Hash comparisons replace character-by-character comparisons of files and text to make sure that data has not been compromised.
- **Password protection.** The hashes are stored instead of the original password. Even if the password files are compromised, the password is not easily recovered by the attacker.
- **Digital signatures and fingerprinting.** The validity of downloaded software can be checked against a hash on the software developer's website.
- **Proof of work in digital currency mining**. Transactions are hashed until a certain number of leading zeros are obtained.

Secure hashing algorithms, from a student's perspective, are relatively complex. The first part of the paper will present the important concepts used in the hashing process. The second part will describe a portfolio of applications that reinforce and explain the secure hashing concepts, and in particular blockchain concepts. We believe that focusing on blockchain technology is a strong motivating force for understanding hashing concepts because of its current popularity.

Blockchain Technology

Blockchain technology is revolutionizing the way that individuals are handling and interacting with cryptocurrency. Many organizations and individuals are trying to get involved in mining, but the tremendous computational and energy demands of mining are reducing the opportunity for success. The decrease in mining success for new entrants is related to the presence of large mining companies and consortia with deep pockets. A mining pool is a group of miners that join forces to combine computing power for monetary gain (Figure 1). Large mining pools increase the chance for successful hashing.

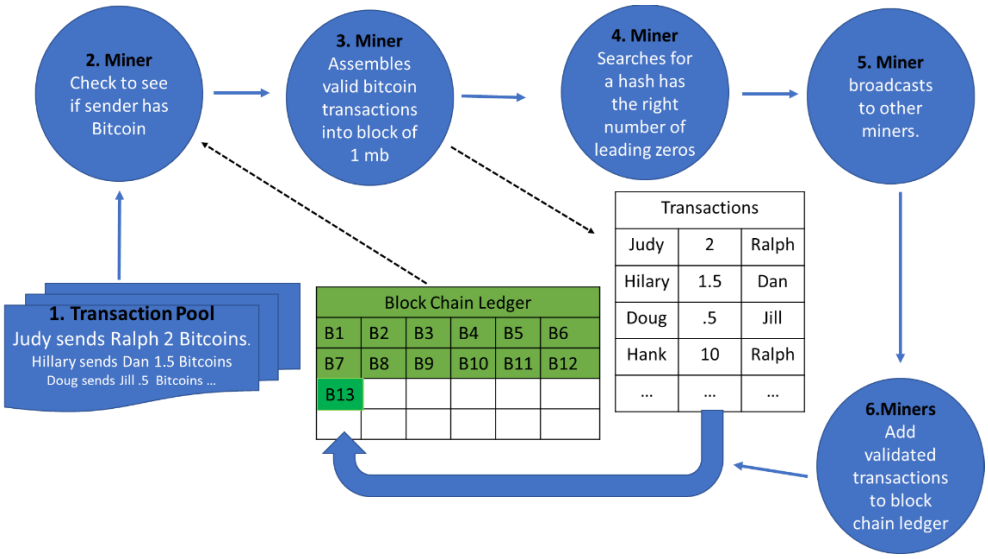


Figure 1: The Mining Process

The underlying hashing algorithm behind blockchain mining can be traced to the 1970s. Sophisticated hashing algorithms emerged in the 1990s. They included the original SHA 1 and MD5 hash algorithms. These algorithms are essential in both computing and blockchain mining because they hold the key to security and authenticity. The authenticity of data is maintained by ensuring the data has not been changed; this is accomplished by using a complex mathematical algorithm. The hash algorithm is also used in mining to make sure that there is no cheating or double spending, and to eliminate the presence of spammers. A spamming attack during the mining process will reduce the efficiency and overall hash rate of the mining process. The mining process is computationally intensive and involves the difficult task of creating a hash with leading zeros. Difficulty in mining increases as the number of leading zeros increases. At this point, Bitcoin miners are required to produce hashes with 17 leading zeros.

Dedicated Hardware Mining

The computational requirements of Bitcoin mining are significant and are a major reason why blockchain implementation does not readily scale. Bitcoin implementation is very time consuming to setup and requires powerful servers and extended infrastructures. Users must not only be concerned with the GPU processing power and RAM, but the power supply, motherboard, and graphics card as well. The computational intensiveness of Bitcoin mining has led to the use of Application-Specific Integrated Circuits (ASICS) that are expensive, loud, have a short life, and produce substantial power costs. It has been estimated that a bitcoin transaction consumes more than 5,000 times more energy than a VIas transaction.

Ethereum is in the process of using a different hash algorithm, Ethash, for proof of work. It is ASIC resistant and permits block verification by a light client. Eventually, GPUs may not be needed to mine digital currency. Rather a grid computing network could be employed involving millions of connected computers to engage in transaction verification.

~~Ethereum appeals to the corporate market through the use of smart contracts, which dictate permission relationships. A smart contract is a way of facilitating, verifying, or enforcing the negotiation of a contracts. On the other hand, Bitcoin enacts in a permission less manner and does not utilize smart contracts.~~

~~Why might one want to implement Ethereum over Bitcoin? One might choose to implement Ethereum because Ether is considered more useful and easier to setup than Bitcoin. You don't have to worry about an infrastructure as much, and you only need to be concerned with having enough GPU (graphics processing unit), processing power, and RAM. To meet these criteria a user can easily pick up another graphics card and have a dual graphics card setup to increase their GPU processing power. Also, users can eliminate the need for more processing power if they do not mine ether. In that case, the user can resort to an online business that will mine coins for them for a fee. With Ether virtually anything can be utilized with blockchain technology.~~

Details on the Hashing Process

A hash function is used to verify that data has not been changed. A hash function is used to map data back to a specific set of data of a predetermined size. If you want to check if any value in a stream of characters has changed, you can check the hash value. A verified hash value indicates that the original data remains untouched. If the hash value does not match the hash value from the original data, then data has been altered or tampered with in some way. This is in part how the double spending problem is countered, and man-in-the-middle attacks are mitigated. It should be noted that checksum computation has some similarities, but checksums are different because they are not unique, whereas the SHA values are essentially unique. And while hash collisions do occur, they are rare with recent implementations of hash algorithms. A simple example of how hashes are generated is shown in Figure 1.

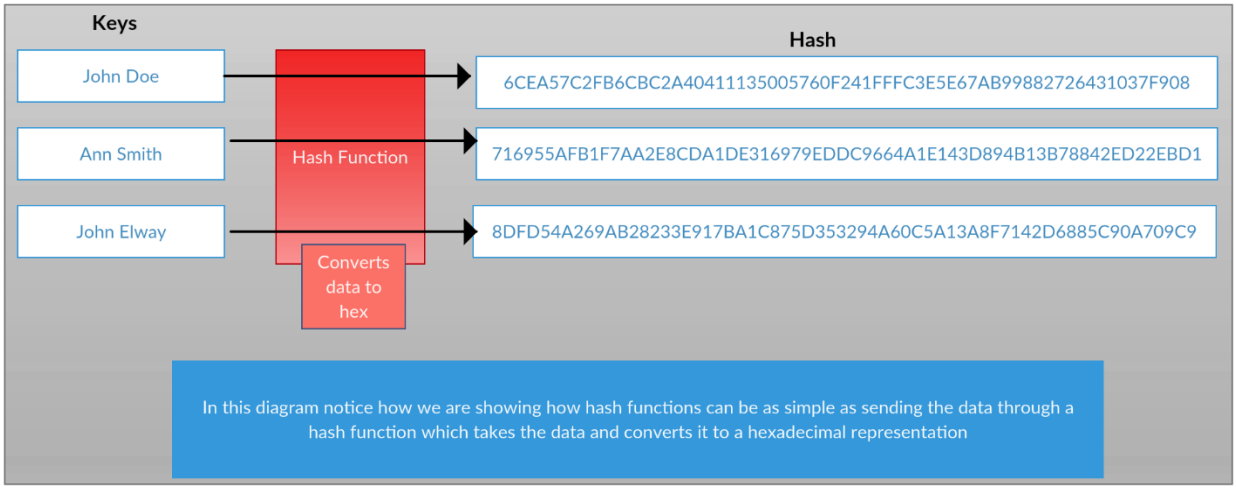


Figure 2: Hash Algorithm Example Developed by the Authors

Hashing algorithms are utilized in computer security. In the case of the blockchain, they are used to ensure that the data blocks have not been modified and act as a verification system for miners. In computer security programs such as antivirus programs, hashes are used to make sure that the applications on your personal computer are not hacked or manipulated in some way. If the hash values of programs and files are different than what is contained in the antivirus database, they will be flagged. The antivirus program checks all installed applications to determine if the hashes in the database match the hashes computed during the virus check. There are many other applications and software that use hashing algorithms to ensure the integrity of data. Below is a diagram showing how hashing works.

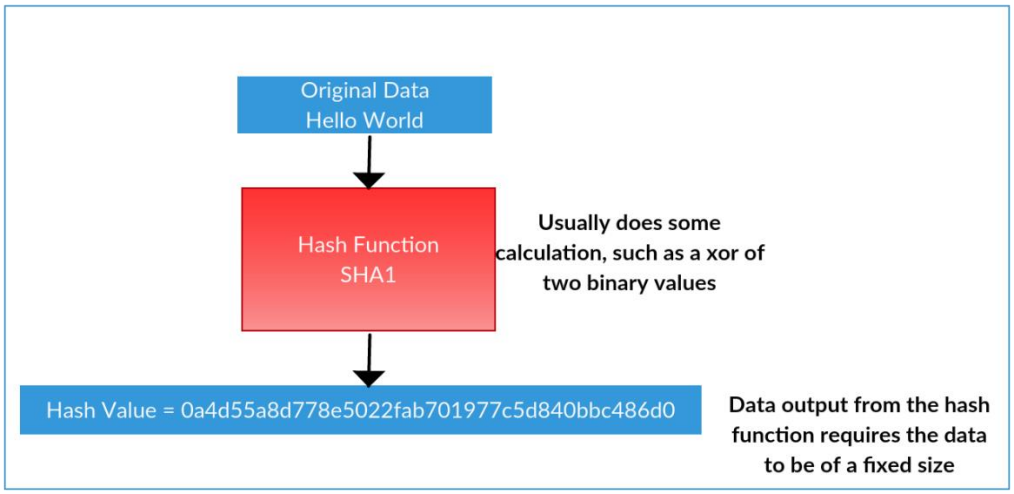


Figure 3: How Hashing Works

Hashing using SHA involves multiple steps, whether it be SHA 224, 256, or 512 algorithms. The first step in SHA is to append padding bits. This means adding a single 1 bit followed by the necessary number of zero bits. For example, if we have 1101 and we require padding of two additional bits, we would get 110100. The key is always to add the zeros padding at the end. The next step is to append the length, which is dependent on the block size. This means we have to use the following rule of $0 \leq k < 512$. The k in this instance should be equivalent to 448, and thus we conclude the formula can be $448 \equiv -64 \pmod{512}$. In any case, the 512 doesn't have to be 512 but can change, depending upon the use of SHA 224 or 256. This step is required to distinguish the empty input from the longer input. The next step is to initialize the hash buffer by representing eight 64-bit registers, each consisting of hexadecimal values (a,b,c,d,e,f,g,h). These values represent the 9th through 16th primes. The formula is as follows: $0 = \lfloor frac(\sqrt{n}) \cdot * 264 \rfloor$. The letter n, in this case, represents the prime number. The next step is to process the message in n-bit blocks. The n-bit depends on the block size. The number of rounds is either 64 or 80, depending on which SHA algorithm you choose to use. In each round, you are to take as an input the buffer value of the previous step and a sum of both the buffer value of the previous step and the previous hash value. The final output result in the last round step is when you have completed each of the above steps. The diagram below provides a summary of all the

SHA steps.

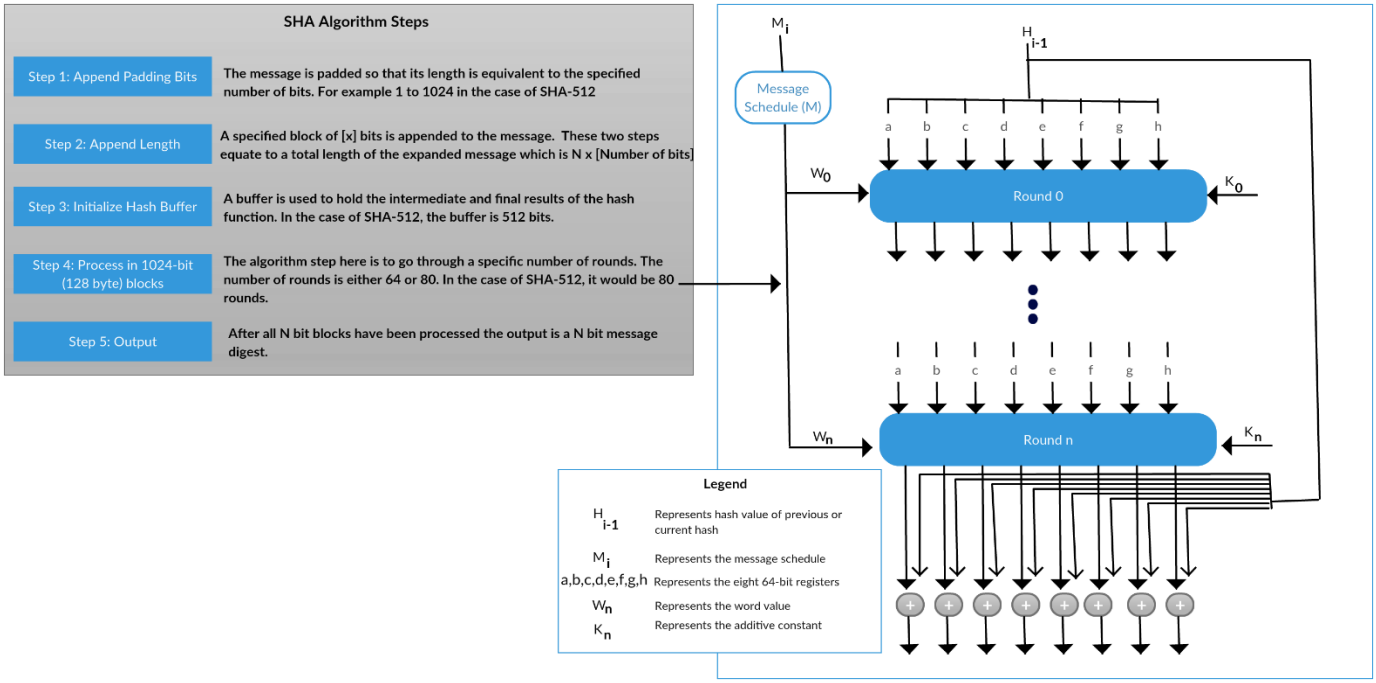


Figure 4: How SHA Algorithm Works (Adapted from Cryptography and Network Security Principles and Practice Seventh Edition)

Hashing Applications for Teaching

A set of three exercises were developed to illustrate hashing concepts. The exercises are coordinated by the instructor, however, class participation is the cornerstone of the exercises. Students should bring a laptop to class, but a smartphone with access to the internet will also suffice. The exercises all use PHP, so they can be run on virtually any device.

Exercise 1: Generating a Nonce

The purpose of the first exercise is to illustrate the output from five secure hashing algoritms and to show how a nonce is used to find leading zeros. Students enter text, submit the text and make observations concerning the hashes for MD2, SHA1, SHA224, SHA256, and SHA384. This program is available at http://104.156.254.129/Exercise1.php.

The first step in Exercise 1 is to tell the students to enter their name. This will generate the five hashes. Then explain that it does not matter how many characters are entered; the hash size for each algoritm will be the same. A fixed size hash will be created. Also explain that it is nearly impossible to take the hash and reverse engineer it to find the original text. This is particularly true of the newer SHA implementations, such as SHA256 and SHA384. You can also note that the SHA256 and SHA384 are more secure than the other algorithms.

The second step in Exercise 1 is to change the first letter of their last name to a lower case. Ask participants to compare the two hashes, which will not be the same.

The third step is to have students enter their name along with the number 1 right after their name. Explain that the number following their name is called a nonce. They will click on the submit button and refer to the MD2 result. If there is no leading zero tell them to put a 2 after their name and see if there is a leading zero for the MD2 hash. This should be repeated until a MD2 hash with a leading zero is generated. On average, this will take approximately 16 iterations. Figure 1 illustrates how the MD2 hash with a leading zero for Sean Sanders with a nonce value of 5 was generated.

Figure 5: Nonce Generation

At this point you can also have students copy a large amount of text from a web page or a file into the text box. Then they submit the text to the hash program. Tell them that the hash is still the same size for each algorithm. Ask them to change any letter in the text to another letter or number, and notice any change in the hashes.

Here is a summary of the steps in Exercise 1:

1. Enter your first and last name into the submit box and click Submit
2. Change the first letter in your last name from upper to lower case and click Submit.
3. Enter your first and last name followed by the number 1 into the text box and then click Submit. If the first character is a zero stop. Capture the screen and save it.
   If the first character was not a zero, keep incrementing the number following your name by one more unit until you generate a hash with 1 leading zero. Capture the screen and save it.
4. Cut and paste some data into the text box and click Submit.
5. Change any letter in the text and Submit.
6. Extra credit: keep adding numbers until you get 2 leading zeros. (Don't try anything beyond 2 leading zeros, for it will take a long time.)

In Bitcoin mining the SHA256 hash is generated by adding a nonce or unique value to the end of the text that is being hashed. We added that number to the end of the name. The goal is to generate a SHA256 hash that starts with zeros. The nonce is the random number that is added to the end of the text being hashed until the desired number of leading zeros are generated. This adding of a random number, or nonce, to generate a hash with leading zeros is what mining is all about.

It takes more time to generate a hash with 4 zeroes in front than 8 zeroes. A hash with one zero requires about 16^1 or 16 attempts. A hash with two zeros requires about 16^2 or 256 attempts, while a hash with 3 zeros requires about 16^3 or 4096 attempts. Right now Bitcoin miners have to generate hashes with 17 leading zeros 16^17 or 2.9514791e+20.

Exercise 2: A Hashing Program that Automatically Searches for a Nonce

This program searches for a nonce for a character string. This algorithm is quite complex because the program has to keep searching until it finds a hash with a leading zero. There are issues related to converting numbers into strings and checking for leading zeros. Dedicated ASICs hardware and GPUs have fine-tuned these operations and are difficult to out-perform. This program is also written in PHP and is available at http://104.156.254.129/Exercise2.html . On small amounts of text this program will have hash rates over 400,000, which are not anywhere near the trillions of hashes per second of ASICs processor.

Here is a summary of the steps for this exercise:

1. Enter your name and the number of Bitcoins you want to give to a friend.
    a. For example: Sean transfers 2 Bitcoins to Matt
2. Enter 5 for the number of leading zeros to generate.
3. Enter SHA512 for the hashing algorithm.
    - ✓ What was total number of attempts?
    - ✓ What was the expected number of attempts?
    - ✓ How long did it take to find the nonce that generated the correct number of leading zeros?
    - ✓ What was the hash rate?

The instructor should ask the class who had the smallest number of attempts and who had the greatest number. He or she should then write them on the board. Also, ask about the hash rate times and ask why they were different. The hash rate is related to the availability of virtual machine resources, but it is still interesting to observe. Tell the class that dedicated mining processors, such as the ANTMINER S9, generate trillions of hashes per second till they find a hash with 17 leading zeros when mining Bitcoin.

Figure 6 presents the inputs and Figure 7 presents the result for “Sean transfers 2 Bitcoins to Matt,” using SHA512 and searching for 5 leading zeros.

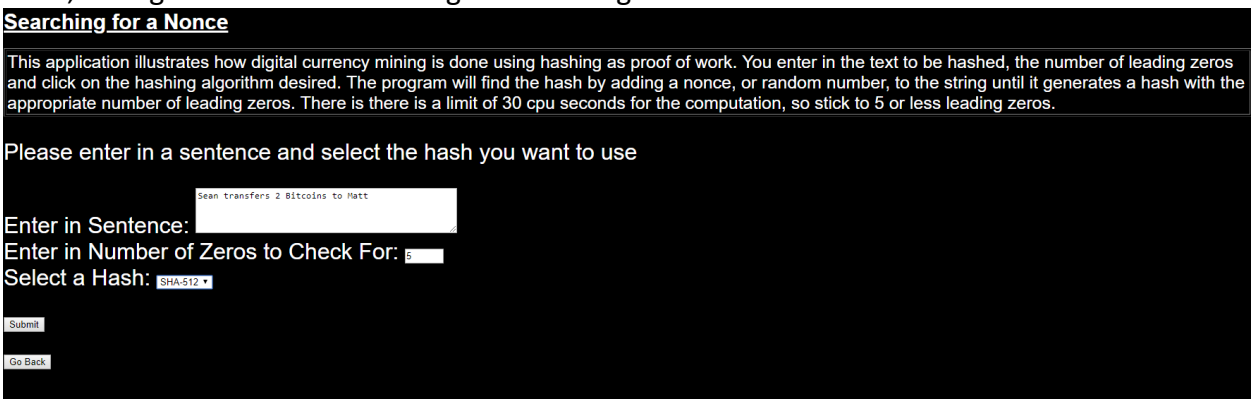


Figure 6: Input

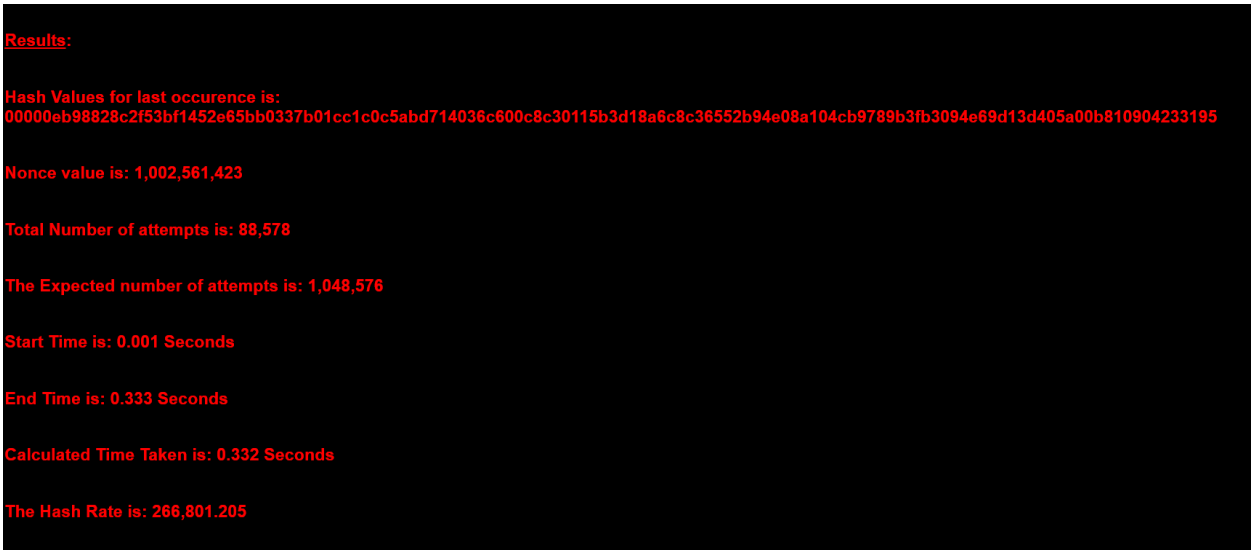


Figure 7: Output

Exercise 3: Mining Simulation

The purpose of this exercise is to illustrate in greater detail the computational demand that is required for using hashing for proof of work. It requires participants to enter in the text to be hashed, along with the number of leading zeros, then to click on the hashing algorithm desired and the number of times to run the simulation. The program will find the hash by adding a nonce, or random number, to the string until it generates a hash with the appropriate number of leading zeros. This program is available at http://104.156.254.129/Exercise3.html It also can be run using a laptop or mobile device with internet access. The simulation can be run with SHA256, SHA512, SHA384 and the older SHA224.

To start the process, students should enter their name and major, check for 4 zeros, select SHA256 for the hash and have the simulation run for 10 iterations. The input is illustrated in Figure 8 and the result of the simulation is illustrated in Figure 9.

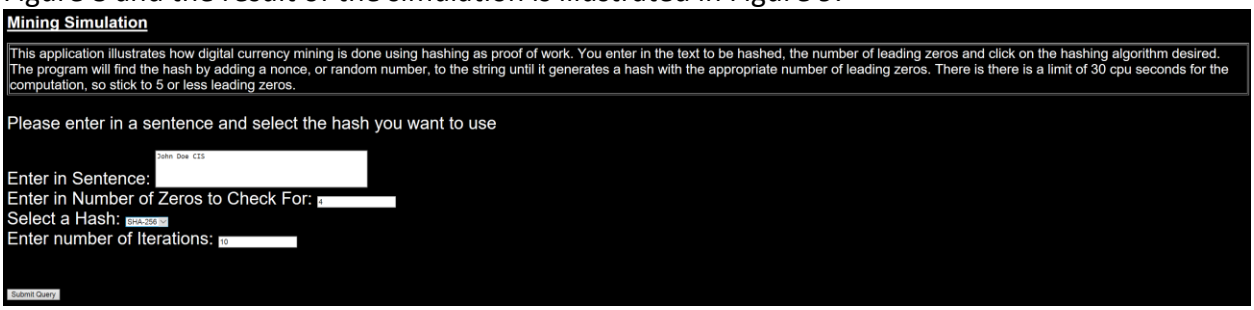
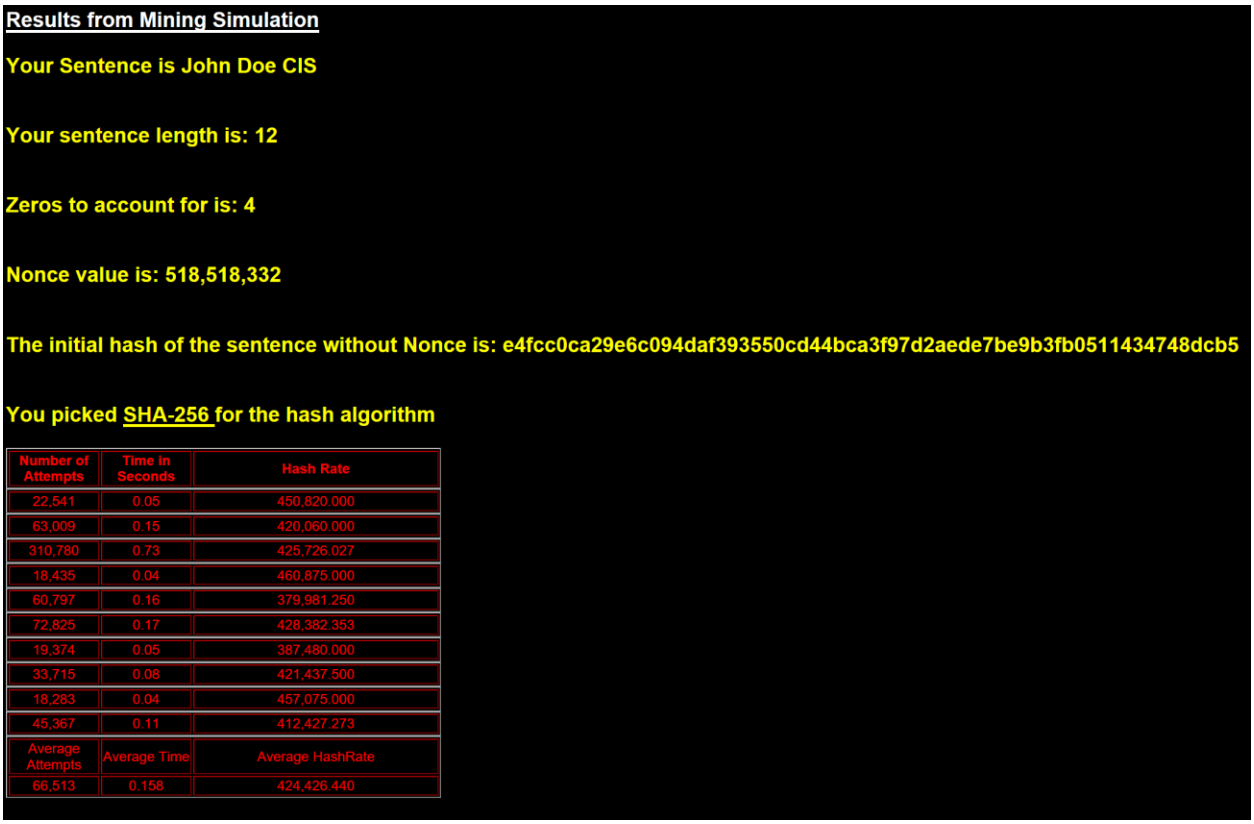


Figure 8: Input



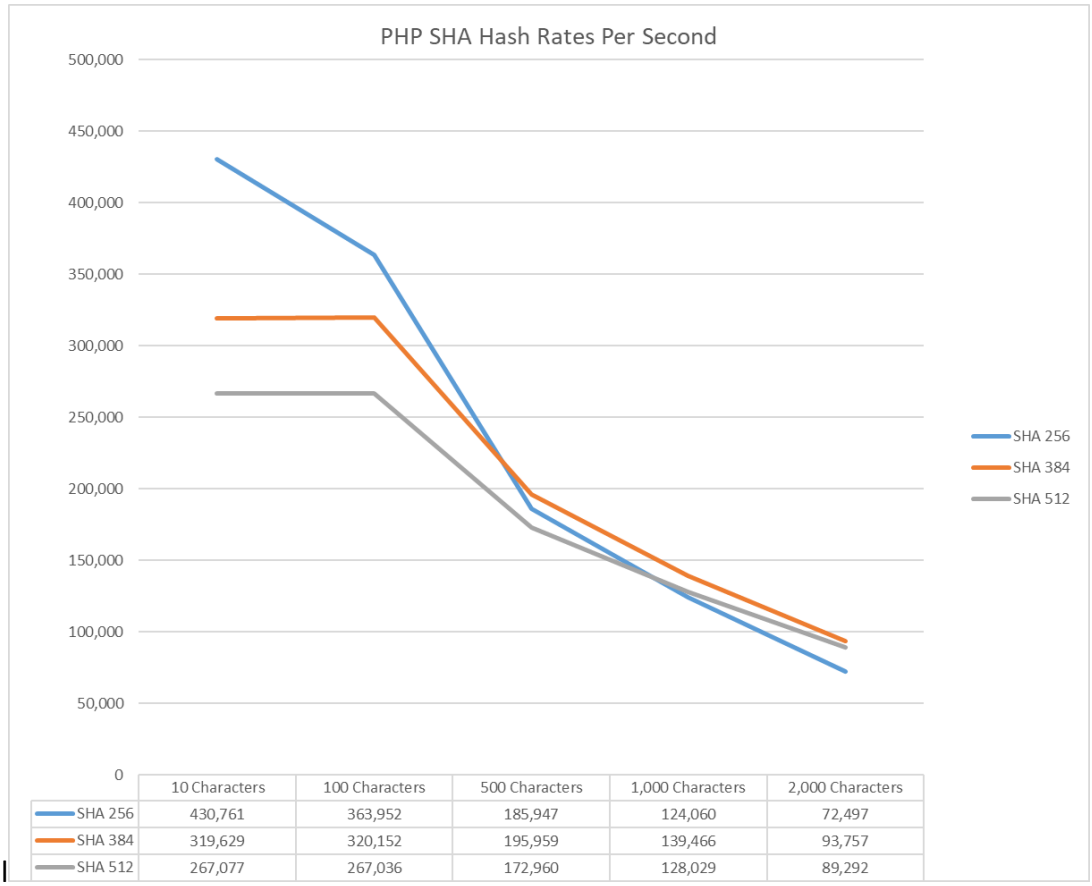
Number of Attempts	Time in Seconds	Hash Rate
22,541	0.05	450,820.000
63,009	0.15	420,060.000
310,780	0.73	425,726.027
18,435	0.04	460,875.000
60,797	0.16	379,981.250
72,825	0.17	428,382.353
19,374	0.05	387,480.000
33,715	0.08	421,437.500
18,283	0.04	457,075.000
45,367	0.11	412,427.273
Average Attempts	Average Time	Average HashRate
66,513	0.158	424,426.440

Figure 9: Output

There are many ways you can use this simulation. It can be used to understand how the number of leading zeros translates to computational intensity. The average number of attempts is a function of the number of leading zeros and is $16^{\wedge n}$ where n is the number of leading zeros.

Students can simulate the performance of the various algorithms and copy the results into a spreadsheet. These results can then be used to write a short paper that discusses the results. Figure 10 was constructed by cutting and pasting the results from running the simulation 100

times for all three hash functions, by varying the number of characters from 10 through 2,000. About 76,838,804 million hashes were used to generate these results. The projected number of hashes to compute was 78,643,200
(100 simulations x 16^4 leading zeroes x 3 SHA algorithms x 4 different character lengths). This is a further illustration that the proof of work simulation is working correctly.



	10 Characters	100 Characters	500 Characters	1,000 Characters	2,000 Characters
SHA 256	430,761	363,952	185,947	124,060	72,497
SHA 384	319,629	320,152	195,959	139,466	93,757
SHA 512	267,077	267,036	172,960	128,029	89,292

Figure 10: Results

Conclusion and Future Direction

This project has explored the importance of secure hash algorithms in a variety of settings, and in particular, digital currency mining. Three exercises were developed to illustrate how secure hash algorithms, are used to increase the security and integrity of data, to understand the various implementations of hashing algorithms and to understand the mining process.

We are currently in the early stages of developing a blockchain simulation called BARTS (Blockchain ART Simulation), where students will be able to participate in simulations of a digital coin for buying and selling drawings. The initial set of lectures will focus on the concepts presented in this paper. The second module, BARTS, will be used to illustrate the following:

market demand concepts, how transactions are processed on the blockchain, the role of smart contracts, how the ledger is updated, and how gas can be used to reduce transaction times. Figure 12 illustrates the basic mechanics of the BARTS process.

Teaching hashing concepts is much easier when the teaching pedagogy is interesting. We think the approach used here establishes a strong foundation for understand hashing concepts, but also creates an opportunity for discussing contemporary blockchain concepts.

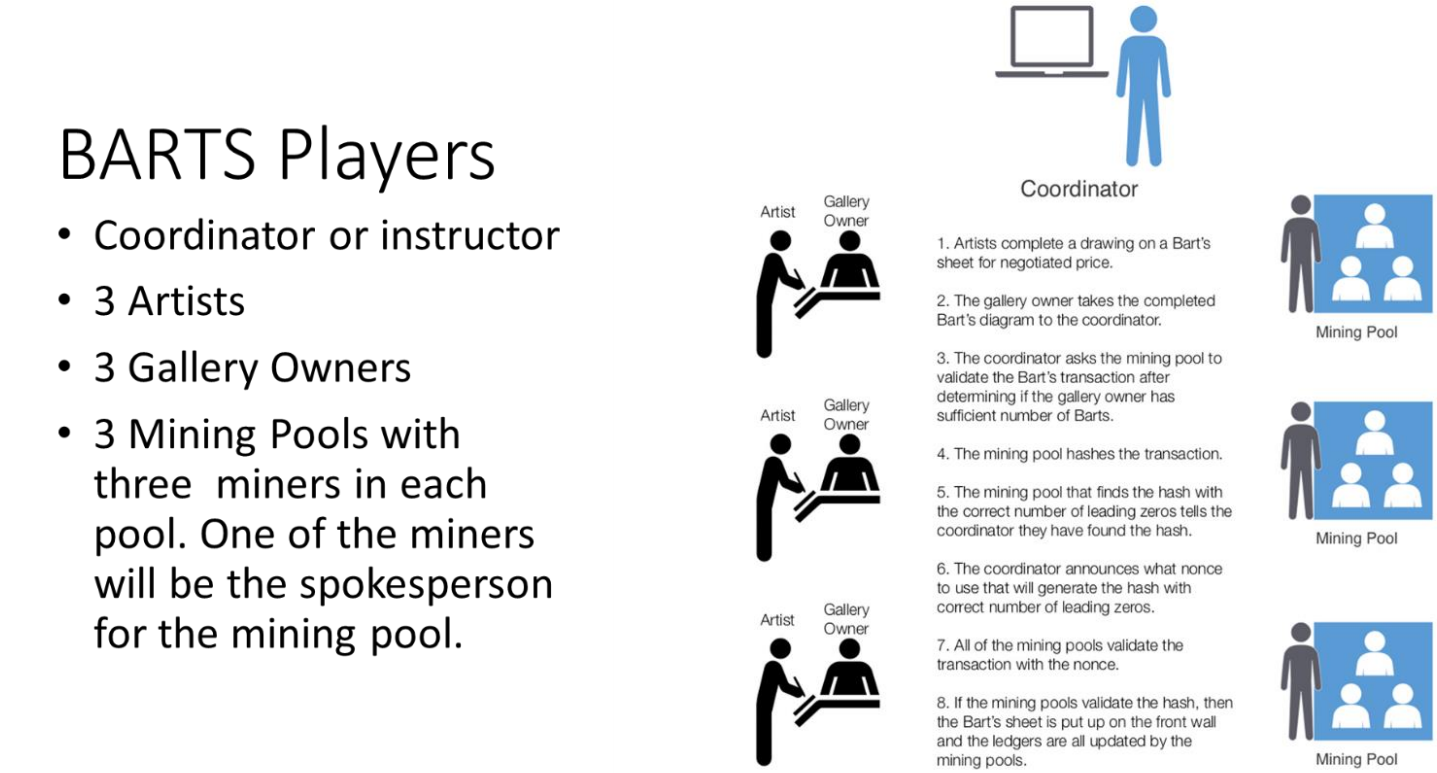


Figure 11: BARTS

#3, 6: A Missing

References:

[1.] Beigel, Ofir. *Bitcoin Mining - What Is It and Is It Profitable in 2018? A Beginner's Guide*, 2017. https://99bitcoins.com/bitcoin-mining-profitable-beginners-explanation/.

[2.] Cachin, Christian. "Blockchain, Cryptography, and Consensus." International Telecommunications Union, March 21, 2017.

[3.] Chang, Iuon, and Tzu Liao. "A Survey of Blockchain Security Issues and Challenges." *INternational Journal of Netowrk Security* 19, no. 5 (2017): 653–58.

[4.] Chaparro, Frank. "Bitcoin Miners Are Making a Killing in Transaction Fees." Blog. Business Insider, August 24, 2017. http://www.businessinsider.com/bitcoin-price-miners-making-killing-in-transaction-fees-2017-8.

[5.] Dev, J. Anish. "Bitcoin Mining Acceleration and Performance Quantification." In *2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*, 1–6. IEEE, 2014. https://doi.org/10.1109/CCECE.2014.6900989.

[6.] Dulat, Michał. "Blockchains: A Brief Introduction." Ragnarson Blog, December 1, 2016. https://blog.ragnarson.com/2016/12/01/blockchains-a-brief-introduction.html.

[7.] Estébanez, Césa, Yago Saez, Gustavo Recio, and Pedro Isasi. "Performance of the Most Common Non-Cryptographic Hash Functions - Estébanez - 2013 - Software: Practice and Experience - Wiley Online Library." Journal. Wiley Online Library, January 28, 2013. http://onlinelibrary.wiley.com/doi/10.1002/spe.2179/full.

[8.] Eyal, Ittay, Adem Gencer, Emin Sirer, and Robbert Renesse. "Bitcoin-NG: A Scalable Blockchain Protocol | USENIX." Journal. Usenix, March 16, 2016. https://www.usenix.org/node/194907.

[9.] Guo, Xu, Sinan Huang, Leyla Nazhandali, and Patrick Schaumont. "Fair and Comprehensive Performance Evaluation of 14 Second Round SHA-3 ASIC Implementations - Semantic Scholar." Journal. Semantic Scholar, 2010. /paper/Fair-and-Comprehensive-Performance-Evaluation-of-1-Guo-Huang/0a1eeac2c74ef77127bbd926b87a13805eb61b6b.

[10.] "Mining Pools - What Are Bitcoin Miners Really Solving? - Bitcoin Stack Exchange." Forum. Stackexchange. Accessed January 9, 2018. https://bitcoin.stackexchange.com/questions/8031/what-are-bitcoin-miners-really-solving/8034.

[11.] Nugroho, K. A., A. Hangga, and I. M. Sudana. "SHA-2 and SHA-3 Based Sequence Randomization Algorithm." In *2016 2nd International Conference on Science and Technology-Computer (ICST)*, 150–54. IEEE, 2016. https://doi.org/10.1109/ICSTC.2016.7877365.

[12.] Pass, Rafael, and Elaine Shi. "Hybrid Consensus: Efficient Consensus in the Permissionless Model," 2016. https://eprint.iacr.org/2016/917.

[13.] Peck, Morgene. "Why the Biggest Bitcoin Mines Are in China - IEEE Spectrum." IEEE Spectrum, October 4, 2017. https://spectrum.ieee.org/computing/networks/why-the-biggest-bitcoin-mines-are-in-china.

[14.] Shirriff, Ken. "Bitcoin Mining the Hard Way: The Algorithms, Protocols, and Bytes." Blog. *Ken Shirriff's Blog* (blog). Accessed January 9, 2018. http://www.righto.com/2014/02/bitcoin-mining-hard-way-algorithms.html.

[15.] Stallings, William. "Cryptography And Network Security." In *Cryptography And Network Security Principles and Practice*, 7th ed., 337–47. Pearson Education Inc., 2017.

[16.] The Ridiculous Amount of Energy It Takes to Run Bitcoin, https://spectrum.ieee.org/energy/policy/the-ridiculous-amount-of-energy-it-takes-to-run-bitcoin