SHA-3 et al.

A close up of a sign

Description automatically generated

Graduate Project 1: The Paper

Eliora Horst

COMP 447: Intrusion Detection & Security

Dr. Corby Schmitz

Loyola University Chicago

INTRODUCTION

While there are countless algorithms used to secure the movement of data across networks, hashing algorithms are the only variety that serve as the digital fingerprints of the network security world. Hashing algorithms achieve this by providing complex, one-way mappings, that make a totally unique hashed message that cannot be reverse engineered to reveal the original data. They are also useful for digital signatures, message authentication codes (MACs), key derivation functions, pseudo random functions, and a variety of other security applications.[[1]](#footnote-1) Hashes are useful for providing authentication of data, and for ensuring data remained unchanged through travel and outside manipulation. The key to hashing, in terms of security, is that every hash is distinct. Every generated digest is unique, and even the smallest of changes to the original data can result in digests that barely resemble the previous iteration.

SHA-3 256-bit Examples

Input: Loyola University Chicago

Output: cf476f6412b09a229e203c6cf2b1981cbd184a085b3a1a2333fa072788690c89

Input: Loyola University Chicaga

Output: 70277db71d7321e161466ce31e3075ed17e1459ded8e29d7fbb36bccb14074bd

[[2]](#footnote-2)

Generated by 'A Security Site.com’

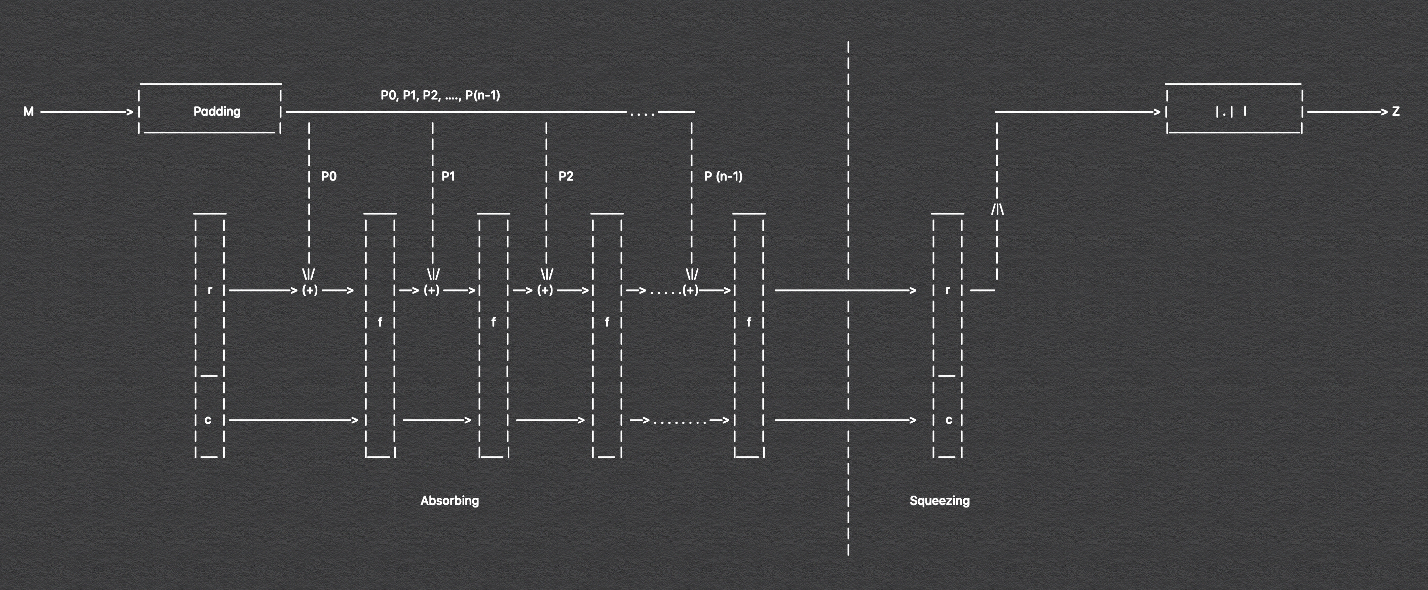
2

SHA-3 is the latest iteration of this hashing technology. Released in 2015, it makes significant improvements to previous versions of the SHA family of hashing algorithms, particularly in regards to collisions. Its internal structure is also significantly different, compared to SHA-1 and SHA-2. It currently utilizes Keccak algorithm, which allows for both variable sized input and output. As of September 2020, there have been no fatal flaws discovered, or any vulnerabilities that could lead to collisions.

HISTORY

The first Secure Hashing Algorithm (SHA) was released in 1993, and was published as a Federal Information Processing Standard (FIPS). This first version was called SHA-0, which was quickly followed by SHA-1. The only difference between the two was that SHA-0 did not include a one bit left rotation.[[3]](#footnote-3) SHA was developed as part of the hashing component of the National Security Administration’s “Capstone Project.” The main purpose of the (now defunct) project was to create an official government standard for public cryptography.[[4]](#footnote-4) Starting in 2005, major flaws in both SHA-0 and SHA-1 were identified, and multiple successful attacks were carried out. As a response, the National Institute of Standards and Technology (NIST) held several public workshops to assess the current standards and status of its own approved standards, and what needed to be done to improve them. From these workshops, NIST decided to develop a completely new cryptographic hash algorithm to be the new standard.[[5]](#footnote-5) In 2007, NIST held the “SHA-3 Cryptographic Hash Algorithm Competition,” which aimed to find the new standard from the public, rather than developing it internally. There were 64 successful projects submitted, which were whittled down to five contenders: BLAKE, Grøstl, JH, KECCAK and Skein. After careful analysis, Keccak was chosen the winner over the others to become the new standard for implementation with SHA-3.[[6]](#footnote-6) While other hashing algorithms are always being developed, SHA-3 and it’s use of Keccak are the current standard for use in the USA government and beyond. In 2015, the SSL industry went through a major transition to make sure all files then using SHA 0 or 1 would be switched over to 2[[7]](#footnote-7).

SHA-3 SPECIFICS



8

Visualization of the SHA-3 algorithmic process

[[8]](#footnote-8)

As stated previously, SHA-3 was implemented using the Keccak algorithm. This new algorithm was chosen to be the standard for several key reasons. Keccak did not supersede any of the previous algorithms in the SHA-2 family of hash algorithms, but rather it was designed to work in tandem with these previous algorithms to double the level of security.[[9]](#footnote-9) “‘SHA-3 is very different from SHA-2 in design,’ says NIST's Shu-jen Chang. ‘It doesn't replace SHA-2, which has not shown any problem, but offers a backup.’”[[10]](#footnote-10) Because of the failings of SHA 1, and the fact that SHA-2 relied on similar mathematics and algorithmic structure which meant that potential for SHA-2 was more inevitable than ever, SHA-3’s completely new and unique internal algorithm and structures made it ever more appealing as the new standard.

HASH FUNCTIONALITY

Hashes, at their core, must strictly adhere to two rules: one, the original message or data cannot be derived from the generated message digest, and two, two messages should never have identical message digests. Hashing algorithms start by taking the original message or data, running it through the hashing algorithm, and delivering a message digest of a specified length. Each successive iteration of SHA algorithms has expanded the length of both the input of the original message and the length of the message digest. Internally, earlier versions of SHA relied on repeated rounds of compression on selected groups of bits, and then combined them at the end to create the digest. The number of rounds of compression and the type of function used to carry out this compression varied from version to version. For example, SHA-1 used 80 rounds of compression to produce a 160-bit message, whereas SHA-256 used 64 rounds of compression to produce a 256-bit message. These earlier developments show that more rounds of compression and lengthier message digests did not always mean the end result was more secure; lengthier message digests also took significantly longer than shorter ones.

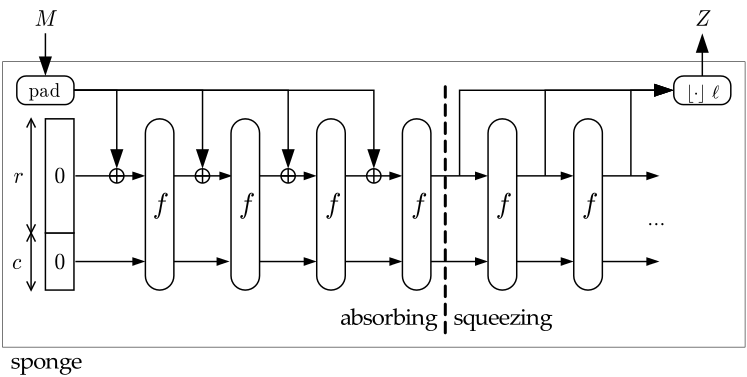
SHA-3 utilizes the Keccak algorithm, which has several features that make it ideal for hash functionality: a large security margin, generally good performance, excellent efficiency, an elegant design, and greater flexibility.[[11]](#footnote-11) SHA-3 works by:

1. Beginning the hashing process by calculating the length of the message/data
2. Padding the data using a padding function, based on the input data length
   1. The padded data will start and end with ones, and all the length in between is filled with zeros[[12]](#footnote-12)
3. Next is the sponge construction, which is detailed below. It is this construction produces the completed hash value.

This process may seem rather simplistic, but its elegant design represents years of hard work and dedication to making the best next generation hashing algorithm.

SPONGE CONSTRUCTION

One of the key features of the Keccak algorithm was its introduction of the new method for chaining, called a “sponge construction,” which can be adjusted and manipulated to provide lower security in trade for faster processing rates, per the users specification.



13

The Sponge Construction

[[13]](#footnote-13)

The Sponge Construction gets its name from the process by which it “absorbs” and “squeezes” data through its processes. This construction takes a binary string of any variable length and returns a string with any specified length. A sponge function is a combination of general hash functionality and stream ciphers.[[14]](#footnote-14)

The sponge construction itself is designed for building the function f with a variable input length, and specified output length, based on a fixed-length transformation of the function operating on a fixed number of bits (bits here being a product of the bitrate and the capacity.[[15]](#footnote-15)

The sponge construction works by first padding the input string, and then cutting that into blocks of size r bits. Then b is initialized to 0, and the sponge construction goes into the first of two phases.

1. Absorption: The input blocks of size r-bit are ⊕ into the first r-bits of the state, using the f function. After all these r-bit sized input blocks have been processed, phase two begins
2. Squeezing: The number of output blocks are specified by the user of the algorithm, and the first of the r-bits of the state are then returned as blocks. Any remaining blocks are truncated.

This sponge construction was developed specifically for use in the Keccak algorithm by the authors. However, their intention is to use this construction beyond the scope of SHA-3. The sponge function is not unique hashing algorithms, but the Keccak algorithm makes use of it for the standard for SHA-3.

KEY DIFFERENCES

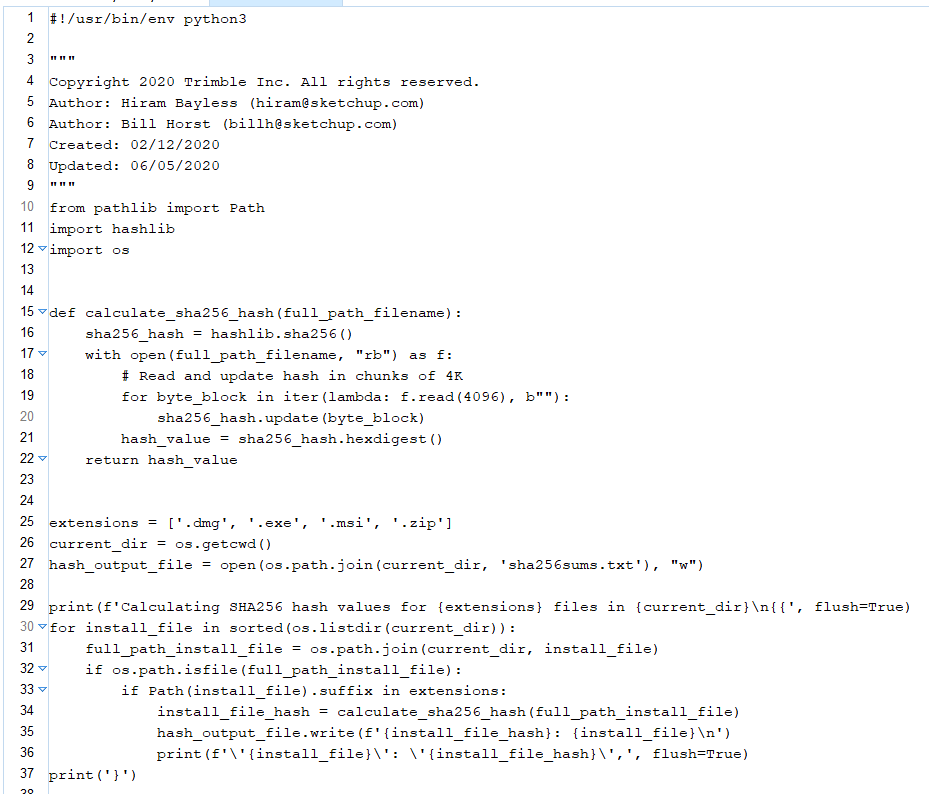
SHA-1 worked by breaking down the original message into 448 bit pieces, which were then padded with 64 bits to make them 512-bit blocks. These blocks were then further broken down into 16 32-bit parts, and those parts were passed through the compression function. This function consisted of 80 rounds of computation, which ended with the algorithm returning a 160-bit output as the final message digest. SHA-3 allows for a variety of input and output lengths to be specified, which makes SHA-3 more flexible than previous iterations. SHA-2 is still in use, and SHA-3 was developed by NIST and the authors of Keccak not as a replacement, but as a compliment.

When SHAs 0/1 were found to have fatal flaws in them, that was the trigger for work on SHA-3, as many files and data at the time were still using SHA-1 certification. Many contemporary browsers and network desktop security programs now issue warnings or outright prevent access to domains still using SHA-1 certification.

REAL WORLD APPLICATIONS

SHA-3 and others have a variety of important applications. Protecting passwords using hash functions means that that services can protect their users passwords even in case of a data breech. If all the stored data only links a users username with a hash value, that means that the users passwords remain safe. Services can perform hash functions on submitted passwords and check them against their stored username + hassed password index, and allow users access to their services if there is a match. Using hashed data is also used for simple and complex data authentication. Hashes can also be used to share access sensative information across networks, or prove to someone else that you know a secret. Another key use for hashes is to identify changes made to messages or data. Original data can hashed, and then hashed again at its final destination. If these two hash values are shown not to be equal, it indicates data corruption, intentional or unintentional. This kind of data can range in size from simple messages to entire applications.

The 3D design software SketchUp utilizes SHA hashes to ensure that the application files they upload to the customer download servers have remained unchanged during the upload process. It is utilized mainly as a validation process. The following Python file is the code for their implementation of their use of the SHA-256 algorithm. The file contains two main components: the implementation of the SHA-256 algorithm inside a function, and the testing of that function.



POTENTIAL ISSUES AND THE FUTURE OF SHA

As attacks on cyber security systems becomes more and more advanced, SHA-2 will inevitably be compromised. However, there are no plans in the works for a migration from SHA-3 to SHA-3 in the near future. While SHA-2 has not yet had any successful attacks, it relies on the same math and similar algorithms as SHAs 0/1, which could lead to it being broken by similar methods in the near future. The attacks on SHA-1 were initially theoretical, but the potential for breaking eventually forced the majority of people to move to SHA-2. For many it was a fortuitus movement, just in the nick of time, as in February of 2017, Google announced there had been a genuine collision attack on SHA-1, and demonstrated that two separate PDF files had the same hash.[[16]](#footnote-16) However, there has yet to be a push for movement from SHA-2 to SHA-3, dispite it’s improvements. This is partially due to a lack of urgency, even though attacks against SHA-2 are becoming more and more sophisticated, and in part due to the fact that many current systems do not have the support to run SHA-3. Some also complain that SHA-3 runs slower than SHA-2 when it comes to software, though when it comes to hardware SHA-3 is the decided victor. Migration from one kind of certification to another is a lot of work, and it would seem wise to start planning a gentle movement to the more secure SHA-3, before the need to becomes imparitive. There needs to be a balance struck between uprooting and moving everything the moment a newer, more secure algorithm comes out, and sticking your feet in the mud and not moving until danger rears its head.

SHA-3 has been described as one of the best hashing security algorithms we currently have. It finds the near perfect balance between diligent security and easy efficiency. Since the cyber security world is constantly evolving and updating, there is no way to guarantee permanent success in security, and it becomes of game of waiting until someone breaks your algorithm in order to spur the next big leap forward in network security. Hashing algorithms are always scurtinized and testing for collisions and other types of volnerabilities, and SHA-3 will inevitably have its moment where it it is blown apart. In the meantime, it remains as the best hashing algorithm out in the wide world of the internet.

REFERENCES

“SHA-3: Keccak, Grøstl, Blake, SHAKE and Skein - The Final Versions | LinkedIn.” Accessed September 25, 2020. <https://www.linkedin.com/pulse/sha-3-shake-keccak-final-version-william-buchanan/>.

Anand, Aditya. “Breaking down : SHA-1 Algorithm.” Medium, January 13, 2020. <https://medium.com/bugbountywriteup/breaking-down-sha-1-algorithm-c152ed353de2>.

———. “Breaking Down : SHA-3 Algorithm.” Medium, January 13, 2020. <https://medium.com/bugbountywriteup/breaking-down-sha-3-algorithm-70fe25e125b6>.

Bertoni, Guido, Joan Daemen, Michaël Peeters, and Gilles Van Assche. “SHA-3 Selection Announcement,” n.d., 1.

Computer Security Division, Information Technology Laboratory. “SHA-3 Project - Hash Functions | CSRC | CSRC.” CSRC | NIST, January 4, 2017. <https://content.csrc.e1c.nist.gov/projects/hash-functions/sha-3-project>.

Grimes, Roger A. “Why Aren’t We Using SHA3?” CSO Online, February 21, 2018. <https://www.csoonline.com/article/3256088/why-arent-we-using-sha3.html>.

Handschuh, Helena. “SHA-0, SHA-1, SHA-2 (Secure Hash Algorithm).” In *Encyclopedia of Cryptography and Security*, edited by Henk C. A. van Tilborg and Sushil Jajodia, 1190–93. Boston, MA: Springer US, 2011. <https://doi.org/10.1007/978-1-4419-5906-5_615>.

Konheim, Alan G. *Hashing in Computer Science: Fifty Years of Slicing and Dicing*. Hoboken, UNITED STATES: John Wiley & Sons, Incorporated, 2010. <http://ebookcentral.proquest.com/lib/luc/detail.action?docID=537352>.

“NIST Releases SHA-3 Cryptographic Hash Standard.” Text. NIST, August 5, 2015. <https://www.nist.gov/news-events/news/2015/08/nist-releases-sha-3-cryptographic-hash-standard>.

Preneel, Bart. “Hash Functions.” In *Encyclopedia of Cryptography and Security*, edited by Henk C. A. van Tilborg and Sushil Jajodia, 543–53. Boston, MA: Springer US, 2011. <https://doi.org/10.1007/978-1-4419-5906-5_580>.

“Q150: What Is Capstone?” Accessed September 25, 2020. <http://x5.net/faqs/crypto/q150.html>.

“SHA-3, Keccak and SHAKE.” Accessed September 25, 2020. <https://asecuritysite.com/encryption/s3?m=The%20quick%20brown%20fox%20jumps%20over%20the%20lazy%20dog>.

Hashed Out by The SSL StoreTM. “What Is the Difference Between SHA-1, SHA-2 and SHA-256?,” November 9, 2018. <https://www.thesslstore.com/blog/difference-sha-1-sha-2-sha-256-hash-algorithms/>.

Team Keccak “The Sponge and Duplex Constructions.” Accessed September 25, 2020. <https://keccak.team/sponge_duplex.html>.

1. “NIST Releases SHA-3 Cryptographic Hash Standard.” Text. NIST, August 5, 2015. <https://www.nist.gov/news-events/news/2015/08/nist-releases-sha-3-cryptographic-hash-standard>. [↑](#footnote-ref-1)
2. “SHA-3, Keccak and SHAKE.” Accessed September 25, 2020. <https://asecuritysite.com/encryption/s3?m=The%20quick%20brown%20fox%20jumps%20over%20the%20lazy%20dog>. [↑](#footnote-ref-2)
3. Handschuh, Helena. “SHA-0, SHA-1, SHA-2 (Secure Hash Algorithm).” In *Encyclopedia of Cryptography and Security*, edited by Henk C. A. van Tilborg and Sushil Jajodia, 1190–93. Boston, MA: Springer US, 2011. <https://doi.org/10.1007/978-1-4419-5906-5_615>. [↑](#footnote-ref-3)
4. “Q150: What Is Capstone?” Accessed September 25, 2020. <http://x5.net/faqs/crypto/q150.html>. [↑](#footnote-ref-4)
5. “NIST Releases SHA-3 Cryptographic Hash Standard.” [↑](#footnote-ref-5)
6. Bertoni, Guido, Joan Daemen, Michaël Peeters, and Gilles Van Assche. “SHA-3 Selection Announcement,” October 2, 2012., 1. [↑](#footnote-ref-6)
7. Hashed Out by The SSL StoreTM. “What Is the Difference Between SHA-1, SHA-2 and SHA-256?,” November 9, 2018. <https://www.thesslstore.com/blog/difference-sha-1-sha-2-sha-256-hash-algorithms/>. [↑](#footnote-ref-7)
8. Hashed Out by The SSL StoreTM [↑](#footnote-ref-8)
9. Bertoni, Guido, Joan Daemen, Michaël Peeters, and Gilles Van Assche. [↑](#footnote-ref-9)
10. “NIST Releases SHA-3 Cryptographic Hash Standard.” [↑](#footnote-ref-10)
11. Bertoni, Guido, Joan Daemen, Michaël Peeters, and Gilles Van Assche [↑](#footnote-ref-11)
12. Anand, Aditya. “Breaking Down : SHA-3 Algorithm.” Medium, January 13, 2020. <https://medium.com/bugbountywriteup/breaking-down-sha-3-algorithm-70fe25e125b6>. [↑](#footnote-ref-12)
13. Team Keccak “The Sponge and Duplex Constructions.” Accessed September 25, 2020. <https://keccak.team/sponge_duplex.html>. [↑](#footnote-ref-13)
14. Team Keccak [↑](#footnote-ref-14)
15. Team Keccak [↑](#footnote-ref-15)
16. Grimes, Roger A. “Why Aren’t We Using SHA3?” CSO Online, February 21, 2018. <https://www.csoonline.com/article/3256088/why-arent-we-using-sha3.html>. [↑](#footnote-ref-16)