

Performance analysis of SHA 2 and SHA 3

Swati Sharma¹, Loveneet¹, Saba Khanum²

¹swati.e11124@cumail.in

²loveneet.e10341@cumail.in

³saba@msit.in

^{1,2}DEPARTMENT OF CSE, Chandigarh University

³DEPARTMENT OF CSE AND IT, MSIT

ABSTRACT

Since its inception in early 2009, Bitcoin has attracted a substantial amount of users and the popularity of this decentralized virtual currency is rapidly increasing day by day. Over the years, an arms race for mining hardware has resulted with miners requiring more and more hashing power in order to remain alive in the Bitcoin mining arena. The hashing rate and the energy consumption of the mining devices used are of utmost importance for the profit margin in Bitcoin mining. As Bitcoin mining is fundamentally all about computing the double SHA256 hash of a certain stream of inputs many times, a lot of research has been aimed towards hardware optimisations of the SHA256 Hash Standard implementations. However, due to attacks on Blockchain, the SHA 256 Hashing algorithm has come under threat. Most of the widely used cryptographic hash functions are under attack today. With the need to maintain a certain level of security, NIST had selected new cryptographic hash function through public competition. The winning algorithm, Keccak will not only have to establish a strong security, but also has to exhibit good performance and capability to run. In this context, we have analyzed SHA-3 finalists along with the used standard SHA-2. The performances of respective algorithms are evaluated by computing cycles per byte. The empirical analysis shows that two SHA-3 finalists viz. Skein and BLAKE perform better which are nearly same as the performance of SHA-2.

INTRODUCTION

The relationship between the internet and the countless millions of users that make up its population and all of the information they share have a, less than positive relationship with those that want access to that information. Much like the relationship between criminals and cops; the relationship is based on constant escalation to find or resolve a weakness.

And in the current state of affairs, the internet, with all of its users and companies, is in no position to patch holes rapidly in comparison to those actively looking to break through them. The problems that cybersecurity has within the realm of blockchain and cryptocurrencies are two-fold. One is with the effectiveness of third-party users in providing this same level of security over their products.

Blockchain is often touted as a world-changing technology and in many ways, it is. However, it isn't necessarily the cure-all panacea for the world's problems that many evangelists would have you believe. Recent events have shown that while the number of cryptocurrency wallets and exchanges is increasing over time. The level of security they provide is wanting when because over the years, millions of dollars in different cryptocurrencies have been stolen.

The security features of many of the important systems across many industries rely on what is known as "security through obscurity" approach in security engineering. The idea in this approach is to keep a system's security mechanisms and implementation secret. However, a major drawback of this method is that the entire system may collapse when someone discovers the security mechanism. One such system arguably is the Society for Worldwide Interbank Financial Telecommunications (SWIFT), which is a secure messaging system used by financial institutions. Some rightly argue that the existing international payment systems such as SWIFT are outdated and of questionable security. For instance, in 2016, the Bangladesh Central Bank lost over \$81 million due to unauthorized transactions that were routed through the New York Federal Reserve Bank using the SWIFT network. Several banks in Southeast Asia were also suspected to be hacked via the SWIFT system. The use of blockchain-based systems probably help avoid the attacks such as noted above. There is arguably no single point of failure or vulnerability in blockchain. It is also important to point out that while bitcoin, the most well-known blockchain application, has had a bad public perception regarding security, the hacking attacks occurred only to other systems that attempted to hold and store bitcoin private keys. It is reported that a bitcoin transaction sent from one party to another arguably has not ever been compromised.

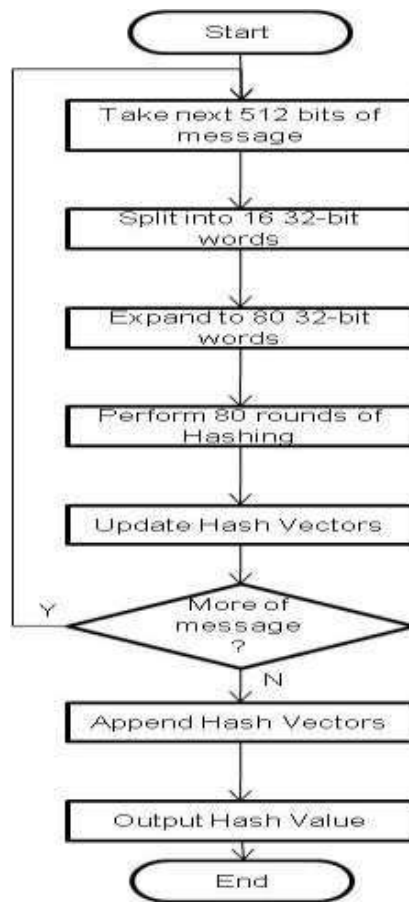


Fig. 1 SHA 2 Flow Diagram

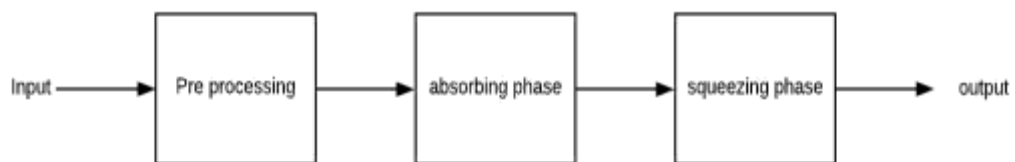


Fig. 2 SHA 3 Block diagram

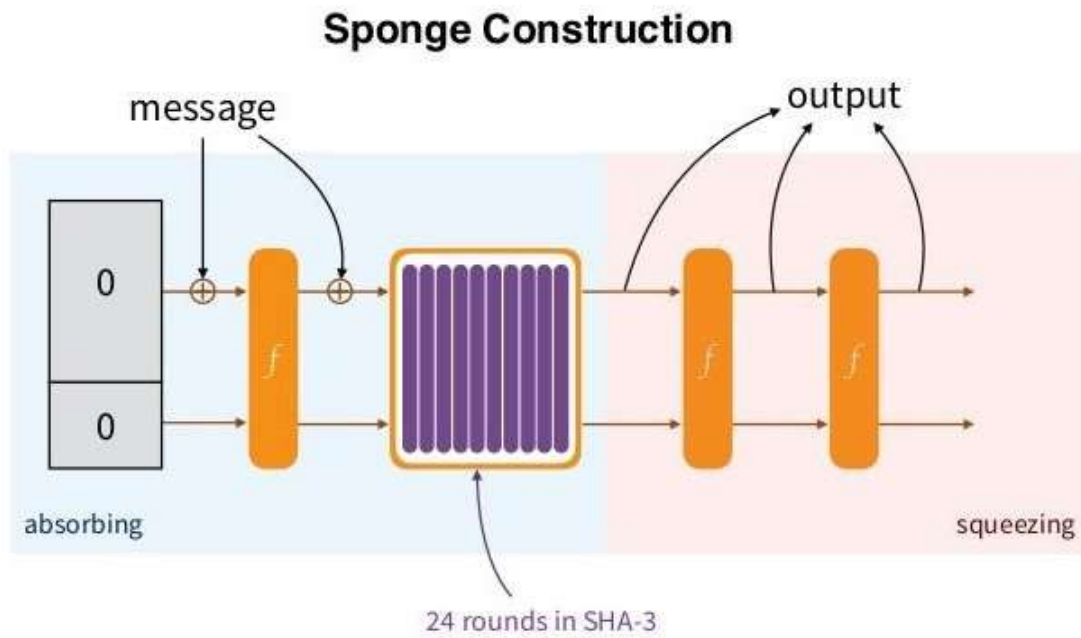


Fig. 3 Sponge construction in SHA 3

IMPLEMENTATION

The SHA256 hashing algorithm operation can be conveniently divided into three distinct operations. They are as follows:

- Pre-processing: Operation that performs padding logic and parses the input message
- Message scheduler: Function that generates sixty-four words from a 16 word input message block
- Compression function: Function that carries out the actual hashing operation of the message-dependent word that comes out of the message scheduler in each round.

SHA-3 uses the sponge construction, in which data is "absorbed" into the sponge, then the result is "squeezed" out. In the absorbing phase, message blocks are XORed into a subset of the state, which is then transformed as a whole using a permutation function f . In the "squeeze" phase, output blocks are read from the same subset of the state, alternated with the state transformation function f . The size of the part of the state that is written and read is called the "rate" (denoted r), and the size of the part that is untouched by input/output is called the "capacity" (denoted c). The capacity determines the security of the scheme. The maximum security level is half the capacity.

```

ADMIN@DESKTOP-3S8E94D MINGW64 ~/Desktop
$ g++ -std=c++0x sha.cpp -o sha

ADMIN@DESKTOP-3S8E94D MINGW64 ~/Desktop
$ ./sha.exe
No input string found, running test using abc.
SHA2 TEST PASSED!

ADMIN@DESKTOP-3S8E94D MINGW64 ~/Desktop
$ ./sha.exe Jerry
7b717950ea1e02ee962fa49549667f329547b833ebce209cd53da66b6b9b264c

```

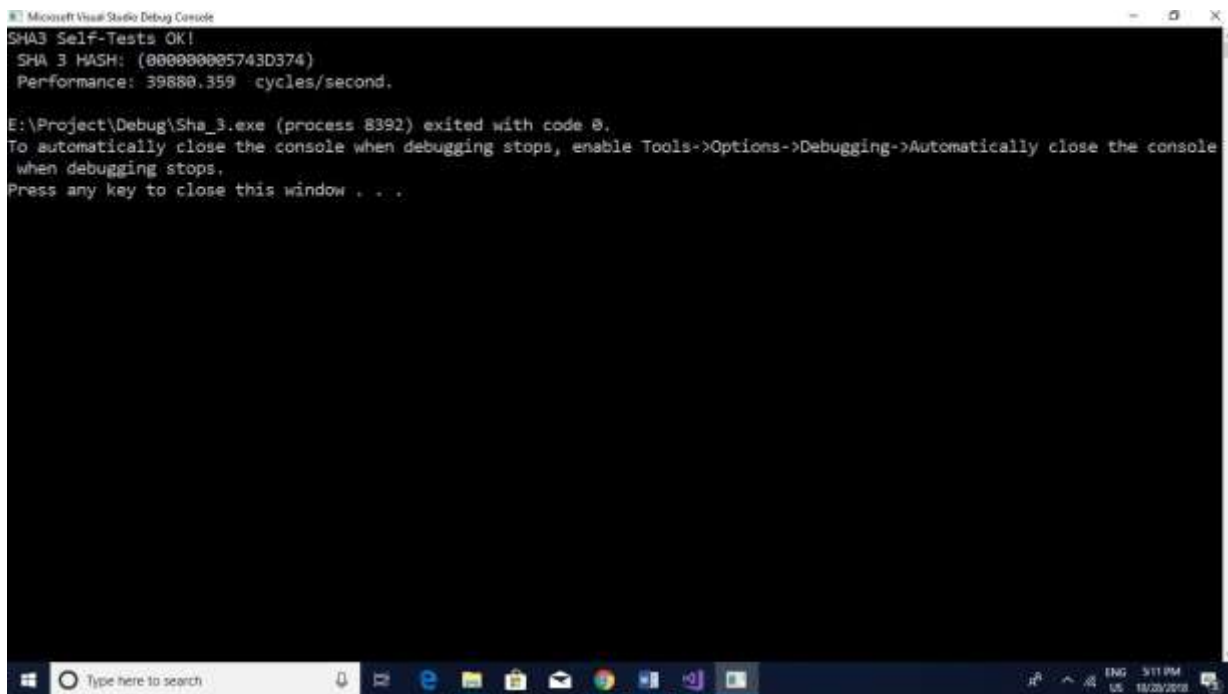
Fig 4. SHA 2 On windows platform

```

vagrant@vagrant:/vagrant$ g++ -std=c++0x sha.cpp -o sha
vagrant@vagrant:/vagrant$ ./sha
No input string found, running test using abc.
SHA2 TEST PASSED!
vagrant@vagrant:/vagrant$ ./sha hello
2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824
vagrant@vagrant:/vagrant$ |

```

Fig 5. Hash generation on Linux environment



```

Microsoft Visual Studio Debug Console
SHA3 Self-Tests OK!
SHA 3 HASH: (000000005743D374)
Performance: 39880.359 cycles/second.

E:\Project\Debug\Sha_3.exe (process 8392) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console
when debugging stops.
Press any key to close this window . . .

```

Fig 6. SHA 3 Implementation

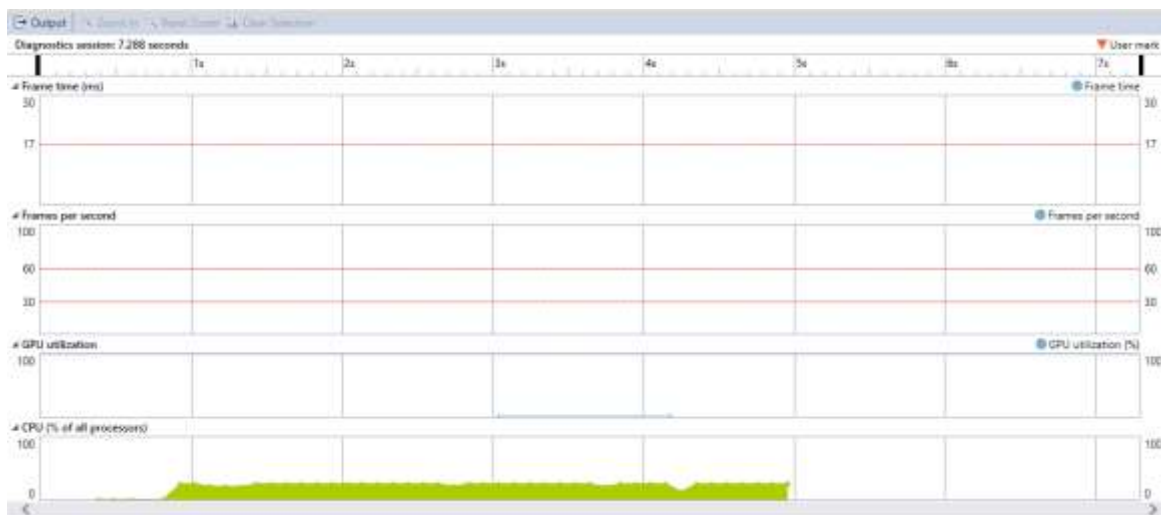


Fig 4 Measuring Performance Parameters

Hash function	SHA 1	SHA 2	SHA 3
Constants Kt number	4	64	16
Size of hash value (n)	160	256	256
Complexity of the best attack	2	80	2
Message size	<2	64	<2
Message block size (m)	<2	64	<2
Message block size (m)	512	512	512
Word size	32	32	32
Numbers of words	5	8	
Digest rounds number	80	64	10

Fig. 5 Functional Characteristics

CONCLUSION

Contrary to the fears leading up to the SHA-3 Competition, SHA-2 has held up well in the face of continued cryptanalysis. The new SHA-3 will need to compete with an existing algorithm (SHA2) that also offers very strong security and performance. Keccak was chosen, not just for its very strong overall security and performance, but because it offers exceptional performance in areas where SHA-2 does not, and because it relies on completely different architectural principles from those of SHA-2 for its security.

Keccak has a large security margin, good general performance, excellent efficiency in hardware implementations, and a flexible design. Keccak uses a new “sponge construction” domain extender, which is based on a fixed permutation, that can be readily adjusted to trade generic security strength for throughput, and can generate larger or smaller hash outputs, as required. The Keccak designers have also defined a modified chaining mode for Keccak that provides authenticated encryption.

FUTURE SCOPE

There might be other algorithms able to determine light weight hash function, which is applicable for the same hash function that can help in the analysis of functions. SHA 3 is one of such functions. Based on keccak algorithm, SHA 3 has very high potential to provide high level of security to blockchaing and cyber security .It is unlikely that we will have a new hash function like SHA 4 until 2013. Therefore keccak is the future and it will be crucial for security application .The study performed in this report will be helpful to predict the use of SHA 3 in future applications. It will also help to secure vulnerabilities in SHA 2 and other similar has algorithms

REFERENCES

- [1] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, SAKURA: a flexible coding, <http://keccak.noekeon.org/Sakura.pdf>.
- [2] R. C. Merkle, A digital signature based on a conventional encryption function, Advances in Cryptology - CRYPTO '87, A Conference on the Theory Applications of Cryptographic Techniques, Santa Barbara, California, USA, 1987, 369-378.
- [3] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, The KECCAK reference, Version 3.0, January 2011, <http://keccak.noekeon.org/Keccak-reference-3.0.pdf>.
- [4] NIST Cryptographic Algorithm Validation Program (CAVP), <http://csrc.nist.gov>
- [5] NIST Special Publication 800-107 Revision 1: Recommendation for Using Approved Hash Algorithms, August 2012, <http://csrc.nist.gov/publications/nistpubs/800-107-rev1/sp800-107-rev1.pdf>

