BOOK CHAPTER | Hashing Algorithms

# Data Hashing Algorithms for Achieving Data Security

[1]Ajayi, Olusola O., [2]Adegbite, Adewuyi A., [3]Aju, Omojokun G. & [4]Falana, Taye F.
[1,2,3]Dept of Computer Science, Adekunle Ajasin University, Akungba-Akoko, Ondo State, Nigeria
[4]Finance and Internal Audit Department, Jumia Nigeria
**E-mails**: olusola.ajayi@aaua.edu.ng; adewuyi.adegbite@aaua.edu.ng; omojokun.aju@aaua.edu.ng
falanatayefedrick@gmail.com

## Abstract

A data is considered useful and helpful only when it is both adequately stored and secured for immediate and future access. Hashing algorithm becomes a necessity where the security of stored data is paramount. The quests to ensure effective security of data led to the proliferation of modified has algorithms aside the fundamental ones. This paper has delivered and discussed the various types of hash algorithms that ensure security of data, as well as technologies for curbing data corruption, pollution and alterations. The main elements of information/data security which are confidentiality, integrity and availability of information/data were also stressed. The paper unfolded the numerous current articles that showcase newer hash algorithms that are subset/modified form/breed of the fundamental hash algorithms.

**Keywords:** data, information, hashing, algorithm, security, data security, information security

## INTRODUCTION

Security in general tone, refers to protection. An author defined it as 'the state of being secured' (Whitman and Mattord, 2012). The field of security is a domain that is generally concerned with protection of assets. It refers to a paradigm, a philosophy and a way of thinking defensively. It aims at protecting and mitigating assets against threats. To prevent vulnerability, a typical security setup deploys the 3Ds techniques: Defense, Detection, and Deterrence.

Information Security is however a streamline term used to describe the protection of the confidentiality, integrity and availability of information/information system/information technology. I.S. consists of the processes and technologies for the protection of information. Putting it straight, information security refers to the processes and technologies that ensure protection and preservation of information.

The three essential elements in information security are:

a) **Confidentiality:** Ensuring that data/information are kept secretive and safe;
b) **Integrity:** Ensuring there is no distortion or corruption of data/information;
c) **Availability:** Ensuring adequacy and appropriateness in the release of the data/information



**Citation** Osa, Edosa (2022). Cyber Security Terminologies and Concepts.
SMART-IEEE-ACity-ICTU-CRACC-ICTU-Foundations Series
Book Chapter on Web of Deceit - African Multistakeholders' Perspective on
Online Safety and Associated Correlates Using Multi-Throng Theoretical, Review, Empirical
and Design Approaches. Pp 253 -260. www.isteams.net/bookchapter2022.
DOI https://doi.org/ 10.22624/AIMS/BK2022-P42

Information Security, according to Andress (2011) can be defined as the protection of information against unauthorized disclosure, transfer, modification or destruction, whether accidental or intentional. Whitman and Mattord (2012) viewed Information Security as a well-informed sense of assurance that the information risks and controls are in balance. According to this study, information security is viewed as a process that ensures that the key characteristics of information (confidentiality, integrity and availability) are actively protected against risks by means of security controls/checks.

Figure 1 shows the various components of information security:

i. Network Security (NS)
ii. Management of Information Security (MIS)
iii. Data Security (CDS)
iv. Policy (P)

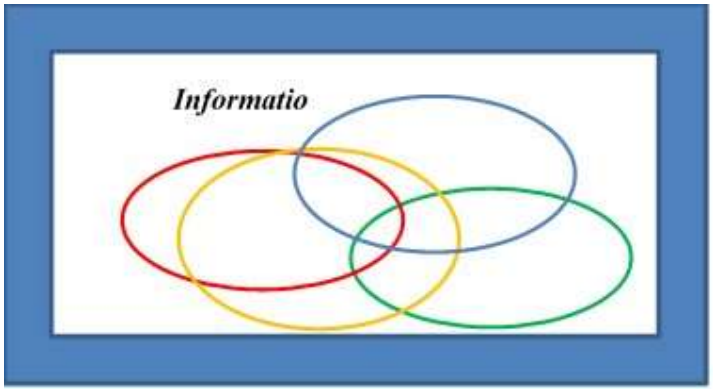


Figure 1: Components of Information Security

As vivid in the diagram, Data Security is one of the components of information security, which refers to the process of protecting data from unauthorized access and data corruption throughout its lifecycle. Sandeep (2014) defined Data Security as Data security as a means of protecting pool of data such as databases from vulnerability, mishandling and unauthorized access and use. Data security means protecting digital data, such as those in a database, from destructive forces and from the unwanted actions of unauthorized users, such as a cyber-attack or a data breach. Data security is the process of securing the data and protecting it from unauthorised and corrupted access. Data security is the measure which is taken to prevent the loss of data through these unauthorised accesses.

Data Security is a process of protecting files, databases, and accounts on a network by adopting a set of controls, applications, and techniques that identify the relative importance of different datasets, their sensitivity, regulatory compliance requirements and then applying appropriate protections to secure those resources. In simple terms, data security is the practice of keeping data protected from corruption and unauthorized access.

### Importance of Data Security

All businesses today deal in data to a degree. From the banking giants dealing in massive volumes of personal and financial data to the one-man business storing the contact details of his customers on a mobile phone, data is at play in companies both large and small. The role and primary aim of data security is to protect the data that an organization collects, stores, creates, receives or transmits. Compliance is also a major consideration.

This is why Data Security is usually seen to be of two-edged sword: Data Protection, and Data Privacy. While Data Protection is described as the law designed to protect personal data in order to empower protection over the data and also avoid abusive use (Costa, 2012), Data Privacy refers to rights established for handling of data, data usage, consent, notice and regulatory obligations (Jeff, 2020).

### Main Elements of Data Security

The core elements of data security: **C**onfidentiality, **I**ntegrity, and **A**vailability, also known as the **CIA** triad (Perrin and Chad, 2012), is a security model and guide for organizations to keep their sensitive data protected from unauthorized access and data infiltration.

i. **Confidentiality** ensures that data is accessed only by authorized individuals;
ii. **Integrity** ensures that information is reliable as well as accurate; and
iii. **Availability** ensures that data is both available and accessible to satisfy business needs.

### Data Security Technologies

Data security is a set of standards and technologies that protect data from intentional or accidental destruction, modification or disclosure. Data security can be applied using a range of techniques and technologies, including administrative controls, physical security, logical controls, organizational standards, and other safeguarding techniques that limit access to unauthorized or malicious users or processes.

Security tools typically provide authentication, encryption, identify attacks, block and filter packets (Ankalarao and Bhavani, 2013). Data security technology comes in many shapes and forms and protects data from a growing number of threats. Many of these threats are from external sources, but organizations should also focus their efforts on safeguarding their data from the inside, too. Ways of securing data include:

i. **Data Encryption**
ii. **Data Masking**
iii. **Data Erasure**
iv. **Data Resilience**
v. **Firewall**
vi. **Authentication**

### Data Encryption:

Data encryption applies a code to every individual piece of data and will not grant access to encrypted data without an authorized key being given. Encryption is the process of translating plain text data (*plaintext*) into something that appears to be random and meaningless (*ciphertext*). Decryption is the process of converting ciphertext back to plaintext. To encrypt more than a small amount of data, *symmetric encryption* is used. A *symmetric key* is used during both the encryption and decryption processes. To decrypt a particular piece of ciphertext, the key that was used to encrypt the data must be used.

The goal of every encryption algorithm is to make it as difficult as possible to decrypt the generated ciphertext without using the key. If a really good encryption algorithm is used, there is no technique significantly better than methodically trying every possible key. For such an algorithm, the longer the key, the more difficult it is to decrypt a piece of ciphertext without possessing the key. It is difficult to determine the quality of an encryption algorithm. Algorithms that look promising sometimes turn out to be very easy to break, given the proper attack. When selecting an encryption algorithm, it is a good idea to choose one that has been in use for several years and has successfully resisted all attacks.

**Data Masking**:
Masking specific areas of data can protect it from disclosure to external malicious sources, and also internal personnel who could potentially use the data. For example, the first 12 digits of a credit card number may be masked within a database.

**Data Erasure**:
There are times when data that is no longer active or used needs to be erased from all systems. For example, if a customer has requested for their name to be removed from a mailing list, the details should be deleted permanently.
**Data resilience**:
By creating backup copies of data, organizations can recover data should it be erased or corrupted accidentally or stolen during a data breach. This data recovery process is also called Data Backup.

**Firewalls:**
Firewalls help you to monitor and control the network traffic. You can restrict access and prevent the spread of malware to your systems.

**Strong User Authentication**
Authentication is another part of data security that we encounter with everyday computer usage. Just think about when you log into your email or blog account. That single sign-on process is a form authentication that allows you to log into applications, files, folders and even an entire computer system. Once logged in, you have various given privileges until logging out.

**Data Hashing Algorithm**

**Algorithm + Hash**
Corman et al. (2002) defined an algorithm as any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output. Yanofsky (2006) defined it as the set of programs that implement or express that algorithm. In computer science, a 'hash' is an algorithm that converts a variable-length value into a constant-length value. The goal of a hash function is to minimize hash collisions, in which two input values will produce the same output value.

A hashing function is any function (·) that can be used to map an arbitrary size of data to a fixed interval [0, m]. Given a dataset containing n data points X = [x1, x2,… xn] ∈ R D, and a hashing function (·), the h(X) = [h(x1), h(x2) ,..., h(xn)] ∈ [0,m] can be called the hash values or simply hashes of data points X = [x1, x2,…xn] ∈ R D (Alfred et al., 1996). In a broader view, the **one way function** that **maps data of variable length to data of fixed length act in such a way that** the **one way function** output cannot be reverse using an efficient algorithm. Also, **mapping data of variable length to data of fixed length** means that the input message space can be “infinite”, but the output space is not.

**Memory Lane**
MD4 was the first widely used dedicated hash function. Developed by Rivest in 1990, it started encountering attacks after which Rivest created MD5, a stronger function in 1992. However, BSD extended DES-BASED crypt to properly support long passwords and permit configurable iteration counts of up to 725 iterations, along with 24-bit salting (Peslyak, 2012). In the early 1990s, attackers targeted passwords in transit by sniffing compromised servers, thereby intercepting passwords transmitted by all the users in a local segment. Developers adapted to the networked computing movement and implemented challenge response pairs, Kerberos, S/Key, and SSH to provide network authentication without exposing protected keys or passwords. With the expansion of the World Wide Web, web application authentication became widely adopted.

This innovation also extended the allowable password length and highlighted the need to secure passwords. In the late 1990s and early 2000s, numerous web developers built application password security around the PHP MD5 module. However, the adopted PHP MD5 hashing function lacks password salting and reiteration (Peslyak, 2012). In 1994, FreeBSD introduced the MD5-based crypt library, which enables the use of long passwords and 1000 iterations of MD5 with up to 48-bit salting. Most Linux distributions adopted the FreeBSD MD5 library by the late 1990s. Early rainbow table attack tools, such as Qcrack and BitSlice, were released around 1997, thereby allowing fast pre-computations of DES hashes while supporting salting designed to attack a dictionary with 4096 possible salts.

As computers and users became connected and networked in the late 1990s, password attacks extended to networks. During this period, non-switched Ethernet was a commonly used technology. Passwords were secured at rest but usually transmitted across networks during operation (Peslyak, 2012). In 1999, Provos and Mazieres proposed ways of building systems in which password security keeps up with hardware speeds. They formalized the properties desirable in a good password system and show that the computational cost of any secure password scheme must increase as hardware improves. They presented two algorithms with adaptable eksblowfish, a block cipher with a purposefully expensive key schedule and BCrypt, a related hash function. The key setup begins with a modified form of the standard Blowfish key setup, in which both the salt and password are used to set all sub-keys.

There are number of rounds in which the standard Blowfish keying algorithm is applied, using alternately the salt and the password as the key, each round starting with the sub-keys state from the previous round. Theoretically, this is no stronger than the standard Blowfish key schedule, but the number of rekeying rounds is configurable; this process can therefore be made arbitrarily slow, which helps deter brute-force attacks upon the hash or salt. Failing a major breakthrough in complexity theory these algorithms should allow password based systems to adapt to hardware improvements and remain secure well into the future.

Thulasimani and Madheswaran (2012) proposed Hash functions that are the most widespread among all cryptographic primitives, and are currently used in multiple cryptographic schemes and in security protocols. The basic design of SHA- 192 is to have the output length of 192. The SHA-192 has been designed to satisfy the different level of enhanced security and to resist the advanced SHA attacks. The security analysis of the SHA-192 is compared to the old one given by NIST and gives more security. The SHA-192 can be used in many applications such as public key cryptosystem, digital sign encryption, message authentication code, random generator and in security architecture of upcoming wireless devices like software defined radio etc.

As of 2012, the most efficient attack against SHA-1 is considered to be the one by Marc Stevens with an estimated cost of $2.77M to break a single hash value by renting CPU power from cloud servers. After some improvements in SHA-1, NSA designed a set of hash functions named SHA-2. SHA-2 was first published in 2001 by NIST. And in August 2002, it became the new Secure Hash Algorithm. Among other benefits, it provides better security than SHA-1 because it has bigger message digest size. Due to this, it is harder to break. The SHA-2 family consists of six hash functions with digests (hash values) that are 224, 256, 384 or 512 bits: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256 (Chaitya and Drashti, 2014).

In May 2014 NIST announced the SHA-3 as the newest proposed scheme for hash functions. SHA-3 is a subset of the cryptographic primitive family Keccak and a cryptographic hash function designed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche, building upon RadioGatún. SHA-3 is not meant to replace SHA-2, as no significant attack on SHA-2 has been demonstrated. Because of the successful attacks on MD5 and SHA-0 and theoretical attacks on SHA-1 and SHA-2, NIST perceived a need for an alternative, dissimilar cryptographic hash, which became SHA-3 (Chaitya and Drashti, 2014).

According to Sriramya and Karthika (2015), BCrypt is currently the secure standard for password hashing. It is derived from the Blowfish block cipher which, to generate the hash, uses look up tables which are initiated in memory. This means a certain amount of memory space needs to be used before a hash can be generated. This can be done on CPU, but when using the power of GPU it will become a lot more cumbersome due to memory restrictions. When BCrypt was originally developed its main threat was custom ASICs specifically built to attack hash functions. These days, those ASICs would be GPUs (password brute forcing can actually still run on GPU, but not in full parallelism) which are cheap to purchase and are ideal for multithreaded processes such as password brute forcing.
FPGAs are similar to GPUs but the memory management is very different. On these chips brute forcing BCrypt can be done more efficiently than on GPUs, but if you have a long enough passwords, it will still be unfeasible. The iteration count is a power of two, which is an input to the algorithm. In 2009 Colin Percival introduced SCrypt which is an improvement upon BCrypt. As noted, the main threat against BCrypt in 1999 was ASICs with low gate counts, but today the threat is FPGAs (Field Programmable Gate Arrays) and BCrypt was not designed to protect against that threat. In the above cases, a function is being provided that developers can put passwords into to get an encoded result. The solutions are improvements upon Morris and Thompson's work for protecting passwords in UNIX systems.

### Structure of the various Hashing Algorithms

**Table 1 shows the structure of the various common hashing algorithms**

| Name | Class | Hash Length |
|---|---|---|
| MD5 | *MD5CryptoServiceProvider* | 128 bits |
| SHA-1 | *SHA1CryptoServiceProvider*<br>*SHA1Managed* | 160 bits |
| SHA-256 | *SHA256Managed* | 256 bits |
| SHA-384 | *SHA384Managed* | 384 bits |
| SHA-512 | *SHA512Managed* | 512 bits |

### More to Hashing, More to Securing Data

*Keyed hash* algorithm can be used to better secure data. Keyed hashes are similar to regular hashes except that the hash is based on a secret key. To verify the hash or to create a fake hash, you need to know that key. The .NET Framework provides two keyed hashing algorithms:

i. **HMACSHA1** This function produces a hash-based message authentication code based on the SHA-1 hashing algorithm. HMACSHA1 combines the original message and the secret key and uses SHA-1 to create a hash. It then combines that hash again with the secret key and creates a second SHA-1 hash. Like SHA-1, the HMACSHA1 algorithm produces a 160-bit hash.
ii. **MACTripleDES** This algorithm uses *TripleDES* to encrypt the message, discarding all but the final 64 bits of the ciphertext.

The hashing algorithms the .NET Framework provides are very efficient and fast, making them useful for many applications. With keyed hashing algorithms, you can send the hash with the data, but you must keep the key secret. Based on the major types of hash functions, several authors have researched and introduced some modified family of hash functions (Knudsen et al., 2005; Eli et al., 2006; Dahmen et al., 2008; Mustafa et al., 2009; Philippe et al., 2012; Jinse et al., 2013; Nasir et al., 2014; Bernstein et al., 2015; Shay et al., 2017; Ajayi et al., 2017; Sevilay et al., 2019).

## Conclusion

This work has presented an overview of the various hash algorithms for achieving data security. Hash algorithm no doubt ensures and preserves data integrity. The relentless efforts of hackers to break the strong room of data through porous or unguarded password mechanism necessitated the quest to come up with better hash algorithms from time to time. Data analysts or application developers should make it a priority to study and apply the most secure data hashing algorithm to ensure the security of stored data and/or password.

## References


1. Ajayi, O. O., and Falana, T. F. (2017). Empirical Evaluation of Data Hashing Algorithms for Password Checks in PhP WebApps using Salt and Pepper. Computing, Information Systems, Development Informatics and Allied Research Journal. Vol. 8(1). www.cisdijournal.net
2. Alfred, J., Oorschot, P. C., and Vanstone, S. A. (1996). Handbook of Applied Cryptography. CRC Press.
3. Andress, J. (2011). Basics of Information Security. Syngress. 1st Edition. ISBN: 9781597496544
4. Ankalarao, K. and Bhavani, V. (2013). Evolution of Security Attacks and Security Technology. International Journal of Computer Science and Mobile Computing. Vol. 2(11), pgs. 270-276
5. Bernstein, D. J., Hopwood, D., Hulsing, A., Lange, T., Niederhagen, R., Papachristodoulou, L., Schneider, M., Schwabe, P., WilcoxO'Hearn, Z. (2015). SPHINCS: Practical Stateless Hash-Based Signatures. In the proceedings of the Annual International Conference on the Theory & Applications of Cryptographic Techniques. 90(56). Pgs. 368-397.
6. Corman, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2002). Introduction to Algorithms. 2nd Edition, McGraw-Hill
7. Costa, L. (2012). Privacy and Precautionary Principle. Computer Law and Security Review. 28(1), 14-24.
8. Chaitya, B. S., and Drashti, R. P. (2014). Secured Hash Algorithm-1: Review Paper. International Journal for Advanced Research in Engineering and Technology. Vol. 2, Issue X, pgs. 26-30
9. Dahmen, E., Buchmann, J., Schneider, M. (2008). "Merkle Tree Traversal Revisited". In the proceedings of the International Workshop on Post-Quantum Cryptography. Lecture Notes on Computer Science. Springer, Berlin
10. Eli, B. and Orr, D. (2006). A Framework for Iterative Hash Functions: HAIFA. In proceedings of the 2nd NIST Cryptographic Hash Workshop. www.csrc.nist.gov/pkt/HashWorkshop2006/program_2006.htm
11. Jeff, P. (2020). Data Privacy Guide: Definitions, Explanations and Legislation. https://www.varonis.com/blog/data-privacy
12. Jinse, S. and Christoph, R. (2013). A Survey of Image Hashing Techniques for Data Authentication in WMSN. In the proceedings of the International Workshop on IEEE International Conference on Internet of Things Communications and Technologies. ISBN: 978-1-4577-2014-7. Pgs. 253-258
13. Knudsen, L. R. (2005). SMASH-Cryptographic Hash Function. In the proceedings of the 12th International Conference on Fast Software Encryption. ISBN: 978-3540-76541-2, pgs. 228-242
14. Mustafa, S. and Ramprasad, J. (2009). Evolving Universal Hash Functions using Genetic Algorithm. In the proceedings of the IEEE International Conference on Future Computer and Communication. ISBN: 978-0-7695-3591-3, pgs. 84-87
15. Nasir, L.M.H., Saleh, E.M., Saadah, D. (2014). Genetic Algorithm. In the proceedings of the 6th International Conference on Computer Science and Information Technology (CSIT), pgs. 23-26

16. National Institute of Standards and Technology (NIST). (2012). Recommendation for Applications Using Approved Hash Algorithms http://csrc.nist.gov/publications/nistpubs/800-107-rev1/sp800-107-rev1.pdf
17. National Institute of Standards and Technology (NIST). (2012). FIPS PUB 180-4, Secure Hash Standard (SHS), US Department of Commerce, Washington D. C., March.
18. Pandey, T. (2016). A Secure Data Transmission over the Cloud Computing: Using Salted MD5. International Journal of Innovative Research in Computer and Communication Engineering. 4(2), pgs. 2784-2790.
19. Perrin and Chad (2012). "The CIA Triad". Retrieved 07/03/2022
20. Peslyak, S. D. (2012). The History of Password Security. http://www.throwingfire.com/the-history-of-passwor-security
21. Philippe, A. and Damel, J. B. (2012). SipHash: A Fast Short-Input PRF. In the proceedings of the 13th International Conference on Cryptology. ISBN: 978-3-642-34930-0, pgs. 489-508
22. Provos, N., and Mazieres, D. (1999). A Future-Adaptable Password Scheme. Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference, Monterey, California.
23. Sandeep, D. (2014). Information and Data Security Concepts, Integrations, Limitations and Future. International Journal of Advanced Information Science and Technology (IJAIST). Vol. 1, No. 2, pgs1-5.
24. Sevilay, V., Donik, K. B., Eren, Y., (2019). A New Approach to Cryptographic Hashing-Color Hidden Hash Algorithm. In the proceedings of the International Conference on Digitization, pgs. 170-173. ISBN: 978-1-7281-3841-1
25. Shay, G. and Nicky, M. (2017). SPHINCS-Simpira: Fast Stateless Hash-based Signatures with Post-Quantum Security. White Paper. https://eprint.iacr.org/2017/645.pdf
26. Sriramya, P., and Karthika, R. A. (2015). Providing Password Security by Salted Password Hashing using BCrypt Algorithm. ARPN Journal of Engineering and Applied Sciences. Vol. 10, N0. 13, pgs. 5551-5556.
27. Thulasimani, L., and Madheswaran, M. (2012). A Novel Secure Hash Algorithm for Public Key Digital Signature Schemes. The International Arab Journal of Information Technology. Vol. 9, No. 3, pgs. 262-267.
28. Whitman, M. E. and Mattord, H. J. (2012). Principle of Information Security. Cengage Learning. 4th Edition.
29. Yan, K., Chen, J., Cao, B., Zheng, Y., Hong, T. (2013). Research on a low conflict flow matching hash algorithm. In the proceedings of IET International Conference on SMART and Sustainable City (ICSSC, 2013), pgs. 234-237
30. Yanofsky, N. (2006). Towards a Definition of an Algorithm. Journal of Logic and Computation. Vol. 21(2).