Hash Functions

Collision – resistant theory
MD 5, SHA 1, SHA 2, SHA 3

Key pointers

DSA
Signing message and Verification Process
HMAC

Example

Hash Value Generation
Signing and Verification

Discussion

Introduction

A hash function is a mathematical algorithm where the input is data of arbitrary length, whereas the output (called a message digest) is a fixed length enciphered text. The message digest, is therefore the hash value, hash code or simply a hash (Sobti & Ganesan, 2012).

The basic requirements for a cryptographic hash function are:

- the input can be of any length,
- the output has a fixed length,
- H(x) is relatively easy to compute for any given x,
- H(x) is one-way,
- H(x) is collision-free.

Cryptanalytic attacks on hash functions, just like with encryption algorithms, seek to exploit some property of the algorithm to perform some attack other than an exhaustive search. Hash functions are however practically easier to attack than encryption algorithms because the attacker does need to assume any secrets and the maximum computational effort required to attack the hash function is only upper bound by the attacker's resources and not user's gullibility.

Examples of cryptographic hash algorithms:

1. The **SHA** (Secure Hash Algorithm) family - published by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS) (Sobti & Ganesan, 2012). This family designates six different hash functions: SHA-0, SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 (*2008 IEEE International Symposium on Industrial Electronics.*, 2008). The first four operate on 512-bit message blocks divided into 32-bit words and the last two on 1024-bit blocks divided into 64-bit words. Bitcoin, the original and largest cryptocurrency (at the time of writing), uses the SHA-256 hash function.
2. The **MD** (Message Digest) family - comprises MD2, MD4, MD5 and MD6 authored by Ronald Rivest for RSA security and was adopted as the Internet Standard RFC 1321 [4].
3. **RIPEMD** (RACE Integrity Primitives Evaluation Message Digest) – a family of cryptographic hash functions based upon the design principles used in MD4 developed

by Hans Dobbertin, Antoon Bosselaers, and Bart Preneel at the COSIC research group at the Katholieke Universiteit Leuven. RIPEMD-160 produces a hash digest of 160 bits (20 bytes).

4. **Whirlpool** – designed by Vincent Rijmen and Paulo S. L. M. Barreto, this hash function is based on a substantially modified version of the Advanced Encryption Standard (AES). Whirlpool produces a hash digest of 512 bits (64 bytes).
5. **BLAKE** – a hash function submitted to the NIST hash function competition by Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C.-W. Phan. It is based on Dan Bernstein's ChaCha stream cipher, but a permuted copy of the input block, XORed with round constants, is added before each ChaCha round.
6. **Curl-P** – a hash function formerly used in IOTA Signature Scheme (ISS). IOTA is a cryptocurrency designed for use with the Internet of Things (IoT) and automotive ecosystems. ISS is based on Winternitz One-Time Signatures but unlike traditional Winternitz, in IOTA users sign the hash of a message. Thus, the security of ISS relies on its cryptographic hash function, which was Curl-P-27.

Name	Block Size(bits)	Word Size(bits)	Output Size(bits)	Rounds
MD4	512	32	128	48
MD5	512	32	128	64
SHA-0	512	32	160	80
SHA-1	512	32	160	80

Explain this table

Collision resistance - this property requires that given a hash function H, it should be computationally infeasible to find two inputs m and m' such that $m \neq m'$ and $H(m) = H(m')$. Due to the fixed size of hash values compared to the much larger – and arbitrary – size of inputs, collisions are expected to exist in hash functions. However, they must be computationally intractable to find.

Collision Resistant Hash Functions (CRHF)

The measure of resistance of a hash algorithm to cryptanalysis is based upon a comparison of its strength to the level of effort required for a brute-force attack. That is, an ideal hash algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort (Sobti & Ganesan, 2012).

Hash-based message authentication code (HMAC)

We can attain data integrity and authenticity of a message using this technique. Well-known hash functions such as SHA-l and MD5 can be extended to produce a HMAC. Also because of the nature of hash functions, a HMAC value can be computed in a much shorter time than a traditional digital signature. We imagine that two parties A and B have a shared secret key k, and ENCk(Msg) and HMACk(Msg) are the symmetric encryption. If A needs to send a message M to B, it first computes ENCk(M) and HMACk(M), and send them to B. B decrypts

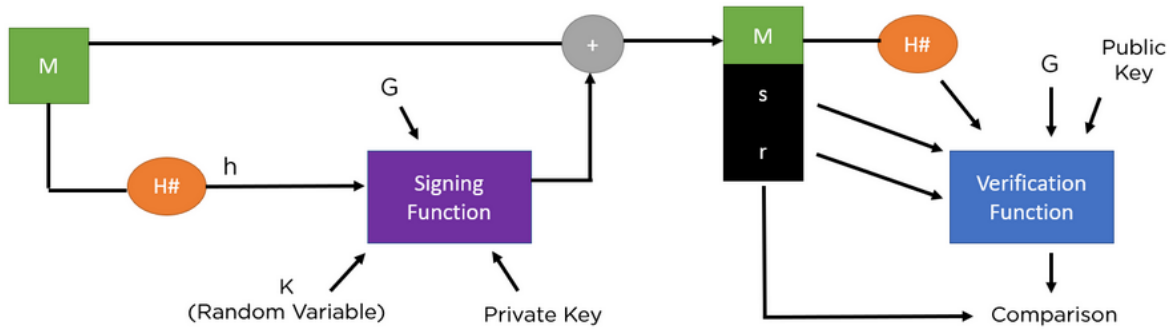
ENCk(M) to obtain message M, and now it calculates the HMAC value of the message. If the calculated value of HMAC resembles the received one, then B accepts the message (Malathi et al., 2012).

DSA - Hash Generation, Signing and Verification Method

Digital Signatures Algorithm is a FIPS (Federal Information Processing Standard) for digital signatures. It was proposed in 1991 and globally standardized in 1994 by the National Institute of Standards and Technology (NIST). It functions on the framework of modular exponentiation and discrete logarithmic problems, which are difficult to compute as a force-brute system.

DSA Algorithm provides three benefits, which are as follows:

- Message Authentication: You can verify the origin of the sender using the right key combination.
- Integrity Verification: You cannot tamper with the message since it will prevent the bundle from being decrypted altogether.
- Non-repudiation: The sender cannot claim they never sent the message if verifies the signature.



[5]

Steps in DSA Algorithm

1. Key Generation

- You first choose a prime number q, which is known as the prime divisor.
- Another prime number, p, is chosen such that p-1 mod q = 0.
- Choose an integer g (1<g<p), satisfying the two conditions, g**q mod p = 1 and g = h**((p–1)/q) mod p
- x is our private key, and it is a random integer such that 0 < x < q.
- y is our public key, and you can calculate it as y = gx mod p.

- Now the private key package is {p,q,g,x}.
- The public key package is {p,q,g,y}.

2. Signature Generation

- It passes the original message (M) through the hash function (H#) to get our hash digest(h).
- It passes the digest as input to a signing function, whose purpose is to give two variables as output, s, and r.
- Apart from the digest, you also use a random integer k such that $0 < k < q$.
- To calculate the value of r, you use the formula $r = (g^k \bmod p) \bmod q$.
- To calculate the value of s, you use the formula $s = [K\text{-}1(h+x \cdot R) \bmod q]$.
- It then packages the signature as {r,s}.
- The entire bundle of the message and signature {M,r,s} are sent to the receiver.

3. Signature Verification

- You use the same hash function (H#) to generate the digest h.
- You then pass this digest off to the verification function, which needs other variables as parameters too.
- Compute the value of w such that: $s*w \bmod q = 1$
- Calculate the value of u1 from the formula, $u1 = h*w \bmod q$
- Calculate the value of u2 from the formula, $u2 = r*w \bmod q$
- The final verification component v is calculated as $v = [((gu1 \cdot yu2) \bmod p) \bmod q]$.
- It compares the value of v to the value of r received in the bundle.
- If it matches, the signature verification is complete.

Having understood the functionality of the DSA Algorithm, you must know the advantages this algorithm offers over alternative standards like the RSA algorithm (Simplelearn, 2022).

Summary

The rapid development of internet of things through Machine Learning and Big Data has significantly increased the volumes of personal data in the modern-day world. The impact of this increasing demand has led to a renewed interest in employing anonymization methods geared towards protecting personal data. By definition, a hash function is a mathematical algorithm where the input is data of arbitrary length, whereas the output (called a message digest) is a fixed length enciphered text. The message digest, is therefore the hash value, hash code, or simply a hash. Generally, a hash function compresses data with specific input value into a shorter output with a fixed numerical value that can mapped in the hash table. The mapped integer value in the hash table takes the form of (key, value) and can be used to calculate the key for any object.

According to Mishra et al. (2021), collision resistance is a hash function's property which dictates that finding two input messages with the same hash value is computationally infeasible. Precisely, this means that the hash function is collision-free. A preimage set is the term given

to any message set that produces the same hash value. It is important to note that a preimage set only exists in theory because it undermines the effectiveness of hash function as a distinctive identifier. For a hash function to be collision-free, the length of its resulting outputs should be at least 160 bits (Berman et al., 2018). In addition, if the hash function is a collision resistant one-way hash function (OWHF), it is considered a collision resistant hash function (CRHF). Validating the definition of collision resistance under certain conditions affects its relationship with preimage resistance.

With many hash functions using numeric or alphanumeric keys, the same hash value can be obtained in infinite messages that are produced. As such, the Message-Digest Algorithm (MD) and the Secure Hash Algorithm (SHA) are types of hash functions developed to address this. Developed in 1991, MD5 encodes an input of any length into a 128-bit message by processing 512-bit blocks of the message through four computation rounds (Pan et al., 2022). The length of the message in MD5 is required to be less than a multiple of 512-bit by 64 bits. MD5 came with new security parameters that slowed down the production speed of message-digest. Ragab et al. (2001) state that when storing sensitive data, such as credit card numbers and passwords, MD5 hashes typically use smaller strings. This led to the discovery that MD5 also undergoes collisions. For instance, in 2005, Neforawati & Arnaldy (2021) found that two digital certificates obtained from diverse public keys had the same MD5 hash. Since then, all MD algorithms are considered unsuitable hashing functions.

The SHA-1 hash function encodes an input of any length into a 160-bit message by processing it in blocks of 512 bits. In any case the input does not match a 512-bit multiple, the algorithm increases its length by padding the message up to the maximum range from the 512-bit multiple (Al-Odat et al., 2019). However, SHA-1 was declared insecure in 2017, and this led to all leading manufacturers of web browsers to stop accepting its SSL certificates. SHA-2 replaced SHA-1 in 2001 and it came with four main modifications: SHA-224, SHA-256, SHA-512, and SHA-384. Both SHA-224 and SHA-256 use a 512-bit block size, but the former produces a 224-bit digest and the latter produces a 256-bit digest. On the other hand, SHA-512 and SHA-384 use a 1,024-bit block size to produce a 512-bit and a 384-bit digest. In 2015, SHA-3 replaced SHA-2 and even though it came with the same hash lengths and variants, its algorithm is more secure (De Guzman et al, 2019).

With reference to digital signatures, hash functions should produce the same output when compared to the input. According to Kumar & Singh (2021), a valid digital signature uses hash algorithm to ensure message authenticated by the sender is not altered, and the sender cannot deny having sent the message (non-repudiation). It has to be authentic, irrevocable, non-reusable, unalterable, and unfalsifiable. The verification process begins with the recipient creating a digest of information encrypted with a private key and unencrypting it using a public key. After that, he compares the created digest with the unencrypted digest. This enables the recipient to use the same hash function to re-compute the message digest before comparing it to the transmitted digest with the aim of verifying that the message did not undergo any modification during the process. A slight modification to the message, such as editing content or adding punctuations usually produces a drastically different digest value (Halevi & Krawczyk, 2006).

As stated by Ahmed & Ahmed (2019), hash mechanisms play a significant role in masking data as long as they have a secret key. In addition, a procedure that make allowances for a secure extraction of the secret key is paramount because it helps in certifying that the completed procedure is irrevocable in nature. In cryptography, the combined use of hash-based message authentication codes (HMAC), secret keys and a meticulous policy of key destruction goes a long way in certifying that the anonymization process is irreversible. This is achievable because it provides both the client and the server with unique private keys that are only recognised by the specific client and specific server. Data holders who keep secret keys used with HMAC may require re-identification after being used to generate pseudonymised data (Berman et al., 2018).

Conclusion

Hash functions are efficiently computable functions that can distribute secret keys uniformly and at the same time condense their input by simulating 'random functions' in a number of ways. The same hash value can be obtained in infinite messages because many hash functions currently use numeric or alphanumeric keys. While they are not infinite, the number of possible outputs in a hash function may be very high. With infinite message space, there is probability of the same hash value being produced. However, since the hash algorithm uniformly covers the whole hash space, all outputs of hash functions theoretically have the same odds of occurrence. Therefore, it can be concluded that all has space values are the hash function's output.

References

Ahmed, A. A., & Ahmed, W. A. (2019). An effective multifactor authentication mechanism based on combiners of hash function over internet of things. *Sensors*, *19*(17), 3663.

Al-Odat, Z., Abbas, A., & Khan, S. U. (2019). Randomness analyses of the secure hash algorithms, SHA-1, SHA-2 and modified SHA. In *2019 International Conference on Frontiers of Information Technology (FIT)* (pp. 316-3165). IEEE.

Berman, I., Degwekar, A., Rothblum, R. D., & Vasudevan, P. N. (2018). Multi-collision resistant hash functions and their applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 133-161). Springer, Cham.

De Guzman, F. E., Gerardo, B. D., & Medina, R. P. (2019). Implementation of enhanced secure hash algorithm towards a secured web portal. In *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)* (pp. 189-192). IEEE.

Halevi, S., & Krawczyk, H. (2006). Strengthening digital signatures via randomized hashing. In *Annual International Cryptology Conference* (pp. 41-59). Springer, Berlin, Heidelberg.

Kumar, S., & Singh, V. (2021, March). A review of digital signature and hash function based approach for secure routing in VANET. In *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)* (pp. 1301-1305). IEEE.

Mishra, N., Islam, S. H., & Zeadally, S. (2021). A comprehensive review on collision-resistant hash functions on lattices. *Journal of Information Security and Applications*, *58*, 102782.

Neforawati, I., & Arnaldy, D. (2021). Message Digest 5 (MD-5) Decryption Application using Python-Based Dictionary Attack Technique. In *2021 4th International Conference of Computer and Informatics Engineering (IC2IE)* (pp. 424-428). IEEE.

Pan, J., Xie, Q., Wang, J., Cui, H., Zhang, J., & Wang, X. (2022). Design and Implementation of Information Digest Algorithm on a Ternary Optical Computer. In *International Conference on Cognitive based Information Processing and Applications (CIPA 2021)* (pp. 788-795). Springer, Singapore.

Ragab, A. H. M., Ismail, N. A., & Allah, O. S. F. (2001). An efficient message digest algorithm (MD) for data security. In *Proceedings of IEEE Region 10 International*

2008 IEEE International Symposium on Industrial Electronics. (2008). I E E E.

Malathi, M., Divya, P. S., Anusha, M., & Rajalakshmi, K. (2012). Multilevel authentication for Vehicular Ad-Hoc Networks with cryptography hash functions. *2012 International Conference on Radar, Communication and Computing, ICRCC 2012*, 303–306. https://doi.org/10.1109/ICRCC.2012.6450600

Simplelearn. (2022). *Cyber Security*. Retrieved from Simplelearn:https://www.simplilearn.com/tutorials/cryptography-tutorial/digital-signature-algorithm

Sobti, R., & Ganesan, G. (2012). Cryptographic Hash Functions: A Review Design of New hash Function using MCC View project. In *Article in International Journal of Computer Science Issues*. https://www.researchgate.net/publication/267422045