APPLICATIONS OF HASHING IN DATA ENCRYPTION

Mahmoud Sharaby – Adham Ali – Ebram Thabet
CSCE 1101- section 5
Department of computer science and engineering

Abstract:

Hashing is one of the cores of computer science. It has many applications in various fields, including cybersecurity, which is used in encryption. It is found in tasks such as protecting passwords, storing in clouds, and validating the integrity of electronic documents and messages. This research examined how hashing has led to great security algorithms and functionalities. It also elucidates the points of strength and weakness in each feature, showing its vast security against cyber-attacks as the main merit and space and time complexity as the prominent demerits. Finally, it introduces quantum computing as a potential solution to the limitations in the algorithms.

Keywords: Hashing, Encryption, Decryption, AES, RSA, SHA256, Bcrypt algorithm, password protection, cloud computing, Digital signatures, quantum computing

1. Introduction:

The world has significantly become technological based, with many tasks that utilize technology daily. For instance, companies use servers to transfer data, people use social media platforms to interact with each other, and many governmental institutions take advantage of technology to store citizen data as well as their confidential information. This has led to concerns about the security and privacy of people's data. Therefore, specialists started to develop methods to make the matter more secure, which is hashing. Hashing is a one-way encryption technique that mainly takes input from the user and converts it to another value. This technique contains hashing mathematical functions that should be deterministic, producing the same output for the same entered data. Additionally, it is infeasible to reverse-engineer the hashed data, which is why it is called a one-way encryption technique. Hashing has numerous functionalities in various fields, such as digital signature authentication, cloud computing, password protection, and other applications. This paper investigates the various applications of hashing that can be used in multiple fields. It explores applications such as cloud computing, digital signature authentication, and password protection, finding the most effective method for securing sensitive data for them by pointing out their strengths and weaknesses. Moreover, this paper mainly highlights the great applicability and versatility of hashing and how it affects the data encryption industry.

2. Motivation:

Encryption is one of the ancient techniques used to protect the sensitive information they are trying to communicate, even if it was leaked to parties that should not get it. For example, one of the main advantages Nazi Germany had in World War II was that they encrypted their communications using a machine that looked like a typewriter. On the other hand, when the Allies decrypted such a code, the Nazi nation lost the war. Technology continued to develop more and more—meanwhile, encryption techniques developed as well, leading to the creation of encryption techniques. One of the leading encryption techniques is hashing. Currently, hashing is ubiquitous; it is found in verification tasks, signature generation, proof-of-authenticity tasks, and many other usages. One of the primary reasons why hashing is so prevalent is that it cannot be reversed back to get the original data, which gives the information high security.

Encryption, in general, and hashing, in particular, are irreplaceable. It is easier to imagine a world with a way of protecting data. For example, how would the powerful companies' databases look without hashing? What would the level of security and privacy be? The situation without encryption techniques is seen when considering the famous cyber-attacks. The Yahoo data breach is an example of data leakage, in which hackers accessed the data of 3 billion users. When this happened, massive concerns about privacy and identity theft have arisen. This incident caused substantial financial issues for Yahoo and significant privacy problems for people. It is worth noting that this case is one of many similar cases that happened before and are happening now.

Combining the fact that hashing is massively used in data encryption and has a lot of benefits and applications with the other scary scenarios of not having hashing and encryption makes this topic extremely amusing as it shows how one of the fundamentals of computer science has evolved and how it has gained its current superiority to other encryption techniques. It also opens new horizons for discovering the ways it can be used.

3. Problem definition

A new issue has surfaced due to the emergence of new social platforms and companies collecting users' data. Thus, there is an increase in data theft, stealing customers' data, resulting in identity theft, and stealing bank accounts. By that, privacy and security had become one of the most focused terms in this era. This paper will tackle this problem by exploring different data encryption applications using hashing.

4. Literature Survey of Methodologies or Algorithms with Analysis and Results:

Digital Signatures:

One of the applications of hashing using data encryption is a digital signature. A digital signature is a mathematical technique that can be used to validate the validity and integrity of electronic documents and messages. It entails utilizing a cryptographic technique to generate a unique digital fingerprint of a document or communication, which may subsequently be

appended as a signature to the document or message. The receiver can then use the same procedure to validate the digital signature, confirming that the document or communication was not tampered with and was sent by the person who claimed it. Digital signatures are essential for ensuring the security and legitimacy of electronic communication and transactions. “Application of Digital Signature for Securing Communication Using RSA scheme based on MD5” is research exploring the use of digital signatures for communication security based on the RSA hashing algorithm. It gives an overview of digital signatures and their significance in verifying the authenticity of documents and messages. The RSA algorithm and the MD5 hashing method can be used to check digital signatures.

The general algorithm for securing communication with digital signatures is based on the RSA scheme and the MDS (Message Digest System):

- Make a public or private key pair for the RSA scheme encryption.
- By using the MDS, generate a hash of the message. This will generate a message representation with a fixed length that will be used for verification.
- Encrypt the hash with the RSA scheme's private key. This will create the digital signature.
- Then send the message as well as the digital signature to the receiver.
- Using the same MDS, the receiver builds a hash of the message.
- Encrypt the digital signature with the RSA scheme's public key, which should submit the message's original hash.
- The last thing, compare the original hash to the received message's hash. If they are identical, this means that the message has not been altered.

Advantages of this algorithm:

- The RSA scheme based on MDS gives complex security through asymmetric cryptography.
- The scheme supports authentication by verifying the sender and guaranteeing that the message did not interfere.
- The scheme provides non-repudiation by guaranteeing that the sender cannot disprove having sent the message.
- The algorithm is very efficient.

Disadvantages of this algorithm:

- The scheme must have secure key management to avoid unauthorizing access to the private key.
- Using hashing can limit the size of messages that can be securely signed.
- It is challenging to distribute securely in some situations because the scheme requires that the receiver has access to the sender's public key.

This is an example of this algorithm in C++:

```
 #include <iostream>

#include <openssl/rsa.h>
#include <openssl/md5.h>

using namespace std;

// Generate a public/private key pair using the RSA algorithm
RSA *generateKeyPair(int keyLength) {
    RSA *rsa = RSA_generate_key(keyLength, RSA_F4, nullptr, nullptr);
    return rsa;
}

// Hash the message using the MDS algorithm
void hashMessage(const unsigned char* message, size_t messageLength, unsigned
char* hashValue) {
    MD5(message, messageLength, hashValue);
}
```

```
// Sign the message using the private key
void signMessage(RSA *privateKey, const unsigned char* message, size_t
messageLength, unsigned char* signature) {
    unsigned char hashValue[MD5_DIGEST_LENGTH];
    hashMessage(message, messageLength, hashValue);
    RSA_private_encrypt(MD5_DIGEST_LENGTH, hashValue, signature, privateKey,
RSA_PKCS1_PADDING);
}

// Verify the signature using the public key
bool verifySignature(RSA *publicKey, const unsigned char* message, size_t
messageLength, const unsigned char* signature) {
    unsigned char hashValue[MD5_DIGEST_LENGTH];
    hashMessage(message, messageLength, hashValue);
    unsigned char decryptedHash[MD5_DIGEST_LENGTH];
    RSA_public_decrypt(RSA_size(publicKey), signature, decryptedHash, publicKey,
RSA_PKCS1_PADDING);
    return memcmp(hashValue, decryptedHash, MD5_DIGEST_LENGTH) == 0;
}

int main() {
    // Generate key pair
    RSA *keyPair = generateKeyPair(2048);

    // Message to be signed
    const unsigned char message[] = "Hello, World!";

    // Calculate the signature
    unsigned char signature[RSA_size(keyPair)];
    signMessage(keyPair, message, sizeof(message), signature);

    // Verify the signature
    bool isValidSignature = verifySignature(keyPair, message, sizeof(message), signature);

    // Output result
    if (isValidSignature) {
        cout << "Signature is valid!" << endl;
    } else {
        cout << "Invalid signature!" << endl;
    }

    // Free memory
    RSA_free(keyPair);

    return 0;
```

(Viega et al.,2002)

Cloud Computing:

Cloud computing is an internet-based technology offering different remote services, such as data storage, software, and hardware, which means applying policies, technologies, and controls to protect data. Therefore, essential principles were made for cloud computing: on-demand computing resources, establishing a pay-as-you-go business model for computing, and information technology services for user use (AbdElnapi et al., 2016). Moreover, several advantages are provided to organizations by cloud computing, for instance, scalability, low cost, and flexibility; however, cloud storage security is a significant problem of cloud computing. Cryptography is crucial for cloud security by designing hash functions, stream ciphers, block ciphers, and ciphers. The main aim of using cryptography is to accomplish the five fundamental information security services: confidentially, integrity, Availability, Authorization, and Non-repudiation. Cryptography has various encryption algorithms; however, the focus will be on three kinds of data encryption: Symmetric Algorithms, asymmetric algorithms, and Hybrid Algorithms related to cloud computing.

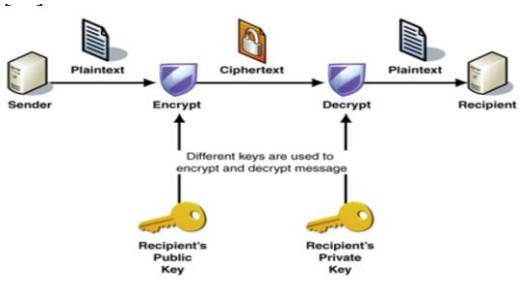


Fig. 2. Symmetric Private Key Cryptography [19].

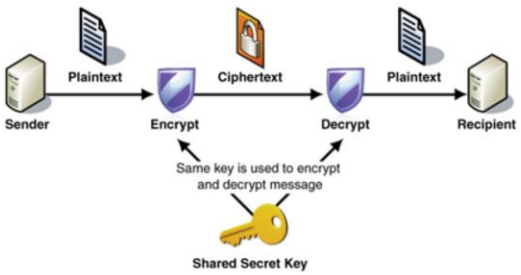


Fig. 1. Symmetric Private Key Cryptography [17].

(AbdElnapi *et al.*, 2016).

By that, AbdElnapi et al. (2016) aim to implement a hybrid encryption algorithm and hash functions to enhance the security of cloud storage by implementing two algorithms, which are Rivest- Shamir-Adleman (RSA) and Advanced Encryption Standard (AES) with a secure hashing algorithm (SHA256) by using Netbeans IDE 8.0.2, JDK 1.7 tool and EyeOS2.5 as a cloud platform on ubuntu14.04 as follow:

A. Generate the public key using a symmetric algorithm (AES)

B. Using an asymmetric algorithm (RSA) to generate the secret key

C. Generating the signature

D. Signature verification and decryption

The recipient should do the following steps to verify the signature and decrypt the file

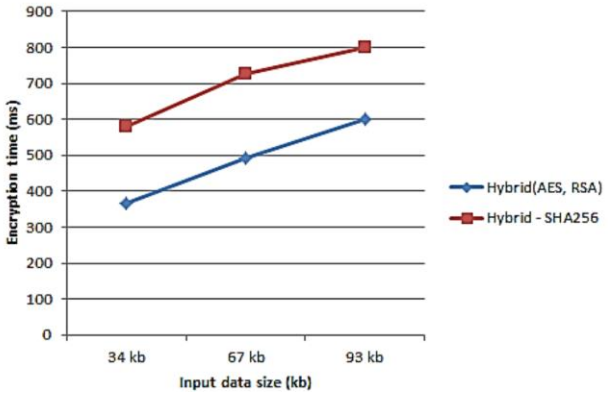
1) Extracts the message digest of the key file information by using the same hash function.

2) Compute the message digest of the information that has been signed

3) If both message digests match, the signature is valid, and he can decrypt the file.

The Results reveal that the new hybrid SHA256 model takes more time to encrypt and decrypt data than the hybrid algorithm, but SHA256 is more efficient and provides more security for data. Because the SHA256 model contains hashing and digital signature concepts, which is more difficult for the intruder to find the plain text from the secret message. Thus, SHA256 provides three fundamental information security services: confidentially, integrity, and non-repudiation. The table and the graph below show the comprising between the new hybrid SHA256 model and the hybrid algorithm.

Input data size (kb)	Time of execution (ms)		
	Hybrid (AES, RSA)	*Hybrid-SHA256*	*Computation overheads with respect to Hybrid (AES, RSA)*
34	365	579	58.63013699 %
67	493	726	47.26166329 %
93	600	801	31.5 %



(AbdElnapi *et al.* ,2016).

Password protection:

Password protection is critical as passwords are used to protect the data of millions of users. Any leakage of passwords may lead to detrimental losses in terms of money and data. That is

why it is crucial to find ways to protect the information, but what processes may be perilous to data privacy? One of the main potentially dangerous processes is transferring data, mainly when it is transmitted via insecure paths. One example is E-commerce, which involves types of businesses that transfer data online (Sriramya & Karthika, 2015). Other examples include social media platforms and all websites with user databases. The typical way of saving data is hugely vulnerable as the data is kept in files as plain text. One of the techniques used to tackle this issue is the Salted Password Hashing Technique, which utilizes the Bcrypt algorithm to encrypt data to an immense size (512 bits) and give the users' data a hash value. The hash functions are helpful in this algorithm as they efficiently localize the data (Sriramya & Karthika, 2015).

How Bcrypt works, enhanced by other algorithms and techniques:

Bcrypt is a function that provides keys for passwords. Bcrypt is implemented with a salt to protect the passwords in storage. The salt is superb in protecting the data from one of the famous attacks called "rainbow table attacks." Bcrypt is based on the Blowfish cipher algorithm, utilizing it to generate the hash using lookup tables stored in the memory. The generated hash string of this algorithm starts with "$2a$" or "2y," indicating that it is in the form of a "modular crypt." The following part of the string is the salt of size 128 bits, then a 192-bit hash value (Sriramya & Karthika, 2015).

Here is one of the implementations of Bcrypt:

```
npm install bcrypt
// app.js
const bcrypt = require("bcrypt");
const saltRounds = 10;
const plainTextPassword1 = "DFGh5546*%^__90";
// app.js
const bcrypt = require("bcrypt");
const saltRounds = 10;
const plainTextPassword1 = "DFGh5546*%^__90";
bcrypt
  .genSalt(saltRounds)
  .then(salt => {
    console.log(`Salt: ${salt}`);
    return bcrypt.hash(plainTextPassword1, salt);
  })
  .then(hash => {
    console.log(`Hash: ${hash}`);
    // Store hash in your password DB.
```

(This implementation is from the link.)

It is evident that Bcrypt, combined with the other algorithms, ensures the security of sensitive data like passwords. Now, it is helpful to trace the workflow of a user trying to login into his account:

1- The user makes an account.

2- Every time the user tries to enter their account, the entered password is converted into a hash value. Then, this value is juxtaposed with the one stored in the database.

3- If they match, the user enters their account. Otherwise, they need access.

Advantages:

1- The generated encryption key is long, making it hard for hackers to decrypt it.

2- It is beneficial to deal with e-force attacks because they try every possible pathway until they reach the target. However, if such a way is used with Bcrypt, the time complexity escalates, which makes this attack type inconvenient.

3- It further provides security against rainbow table attacks since the salt is random.

4- Similar to other hashing algorithms, Bcrypt generates an entirely different hash when the input changes even by one bit. This makes it irritating for hackers to understand the pattern.

Disadvantages:

1- It is sometimes hard to implement and understand for beginners.

2- It takes space from memory before generating the hash, which may result in unsatisfactory performance.

3- If the hash is not long enough, it may be subject to brute-force attacks using FGPAs that have recently become cheap and more powerful.

Potential solution:

Analyzing the algorithms mentioned above and techniques shows that the main drawback of hashing is the time complexity. One of the leading technologies that are increasingly time-efficient is using quantum computers. Quantum computers have qubits as the corresponding structure to bits in classical computers. The qubits have the feature of superposition. This means that qubits can be in multiple states at the same time. Applying this to the hashing algorithms guarantees that the immense time used to run each of the tasks of hashing will be parallelized and distributed because the qubits allow multiple states simultaneously.

5. Conclusion:

From what was read and investigated, data encryption using hashing is a fundamental key to data security today. For instance, hashing provides powerful ways of protecting passwords in

password protection. These ways are secure against attacks, such as the rainbow table and brute force attacks. Nonetheless, it may need to be faster to implement. It is also crucial to consider the hash size to prevent attacks using FGPAs. We reached a pattern protection method for data online by combining digital signature and hashing. However, it takes more time than usual to encrypt and decrypt the data.

It is worth mentioning that even though hashing is robust, it still needs some improvement in terms of space and time complexity. One way is to implement them using quantum computers as they take advantage of the superposition of the qubits.

References:

AbdElnapi, N. M., Omara, F. A., & Omran, N. F. (2016). A hybrid hashing security algorithm for data storage on cloud computing. *International Journal of Computer Science and Information Security (IJCSIS)*, *14*(4). https://www.researchgate.net/publication/303842608_A_Hybrid_Hashing_Security_Algorithm_for_Data_Storage_on_Cloud_Computing

Fashoto, S. G., Gbadeyan, J. A., & Okeyinka, E. A. (2010). Application of Digital Signature for Securing Communication Using RSA Scheme Based on MDS. https://bit.ly/3oIS0sb

Paar, C., & Pelzl, J. (2010). Understanding cryptography. https://doi.org/10.1007/978-3-642-04101-3

Sriramya, P., & Karthika, R. A. (2015). Providing password security by salted password hashing using the bcrypt algorithm. *ARPN journal of engineering and applied sciences*, *10*(13), 5551-5556. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=9d82620f0866ec773b12c30f90d70b74661f972f

Stallings, W. (2011). Web-based security protocols. *Encyclopedia of Cryptography and Security*, 1383–1384. https://doi.org/10.1007/978-1-4419-5906-5_669

Viega, J., Messier, M., & Chandra, P. (2002). Network Security with OpenSSL. O'Reilly Media. Chapter 10: Public Key Cryptography, pp. 185-194. https://www.oreilly.com/library/view/network-security-with/059600270X/