Symbiosis

Research Article | Journal of Computer Science Applications and Information Technology | Open Access

Received: May 04, 2017
Accepted: May 31, 2017
Published: June 23, 2017

Using Cryptology in Enforcing Database Security Traveling Through Networks and Cloud Computing

Ihssan Alkadi*

Independent Researcher, Prairieville, LA 70769-3497, USA

***Corresponding author:** Ihssan Alkadi, Independent Researcher, Louisiana State University, Prairieville, LA 70769-3497, USA, E-mail: ialkadi@gmail.com

1 Abstract

This paper discusses what cryptography is, the practice of using cryptography, and different cryptographic algorithms and their strengths and weaknesses. Cryptographic measures are often the last line of security in a potential security breach and it is important to implement into an organization's security grid. However, using a cryptographic system does come with risks. This paper will go over those risks as well as what makes a cryptographic system strong and what makes them weak. Using a weak cryptographic system can be worse than using no cryptography at all, and we will look into some of the potential attacks on a weak cryptographic system.

2 Introduction

Cryptography is the art of "extreme information security". It is extreme in the sense that once treated by an algorithm, a message (or database field) is expected to remain secure even if the adversary has full access to the treated message [1]. Other security techniques are designed to keep users away from the information and typically come with complicated procedures on authorization roles. Cryptography assumes that every user has full access to all data and if implemented properly, the data still will remain secure.

Cryptography is more commonly known as the scrambling of a data message into a random sequence of bits. For instance, when a user creates an account for their online banking system they use the following credentials: "BobbyBrown51" as the user name and "Bond007" as the password. After creating the account, a cryptographic algorithm is used on the password before it is stored in the database, leaving the password to be stored as an indiscernible string of data like "0397HJ61FBAX9731F". In this case, if the data was breached by a hacker or just by a bank employee who had read-only access working with user accounts, the password for BobbyBrown51 would not be compromised. This is the simplest and most basic example of cryptography, but this is why cryptology is known as the "last line of defense". Note that these algorithms can be applied to more than just passwords and can be applied to usernames, messages, numbers, and all sorts of data.

2.1 Terms

These are a few terms which are used frequently throughout the paper and can get confusing, they are described below:

plaintext: the data in the decrypted, readable form.

encryption: The scrambling of plaintext by a cryptographic algorithm. decryption: The unscrambling of encrypted data.

cipher: Reverse algorithm used on encrypted data to eventually find the plaintext. ciphertext: The scrambled output of a cipher. Converted to plaintext using the key.

key: Unique code that is kept secret and protected, used during the encryption and decryption of data.

3 Cryptographic Algorithms

There is a basic rule in cryptography known as Kerckhoffs's Principle. This rule states that a cryptosystem should be secure even when everything about the system is public knowledge, except for the key. Below are three different categories of cryptographic algorithms, each plays an important role in most cryptographic systems. They are symmetric-key, public-key, and hashing.

3.1 Symmetric-key

Symmetric-key is an encryption system in which the sender and receiver of a message share the same, common key [3]. The same key is used to encrypt and decrypt the message. This is contrast to a public-key system where each individual would have two keys. Symmetric-key systems are faster and simpler than a public key system, but their major disadvantage is that the two parties must exchange the key in some type of secure way. If this key were intercepted, the ciphertext could be ciphered by a perpetrator. Remember, according to Kerckhoff's Principle, the most important thing to keep secure in a cryptographic system is

Citation: Ihssan Alkadi (2017) Using Cryptology in Enforcing Database Security Traveling Through Networks and Cloud Computing. J Comp Sci Appl Inform Technol. 2(3):1-4

the key. Below is a figure of how the symmetric-key works.

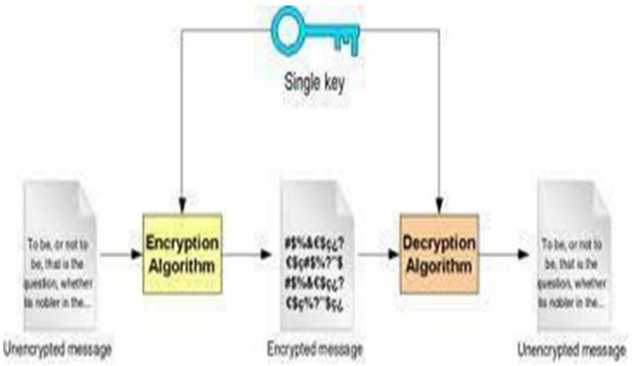


Figure 1: Symmetric-key system [4]

Above, in Figure 1, you can see the basic layout of a symmetric-key system. Imagine this real world example where Tom is sending an important message to Joe, perhaps user account information or banking information. When Tom sends the message, it is first encrypted and then sent to Joe as ciphertext. While this is happening, the key is also sent over a different and secure line. The line that the ciphertext is sent over doesn't necessarily have to be sent secure because the ciphertext is useless without the key. Once Joe receives the ciphertext and the key, the message can be decrypted and he would be able to view the message. Two famous ciphers, Data Encryption Standard (DES) and Advanced Encryption Standard (AES), both use symmetric keys [1]. Symmetric-key systems are efficient at ciphering small and large amounts of data but the biggest risk is key management, because the key has to be sent every time data needs to be deciphered.

3.2 Public Key

As stated before, in comparison to a symmetric-key system, a public key system (asymmetric system) has two keys. One key is known as the public key, which can be shared with the general public. The second key is known as the private key, which must remain a secret. Typically, the public key is the encryption key and the private key is the decryption key, but this is not always the case. Well-known asymmetric algorithms include RSA, ElGamal, and Diffie-Hellman [1]. This type of system is slower and more complex than symmetric-key but is generally more secure. It is commonly used to send smaller amounts of data. Above is a figure of how public key works.

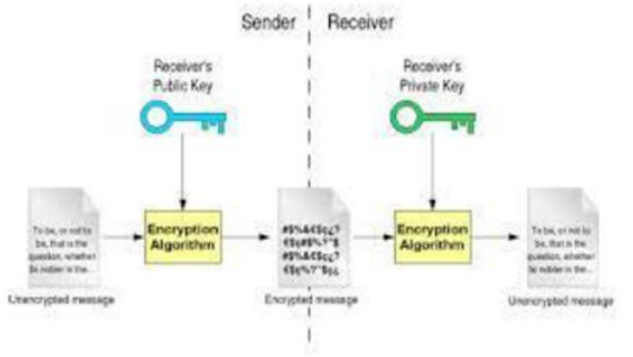


Figure 2: Public key system [4]

Above, in Figure 2, you can see the basic layout of a public key system. Imagine the same real-world example used in Figure 1 where Tom sent a secure message to Joe. In this case, we will assume that Tom and Joe have already been supplied with both of their private keys. When Tom sends the message the message will be sent along with the public key. Both the message and the public key will be encrypted using Tom's private key. When Joe receives the message, his private key will be used in conjunction with the received public key, and he will then be able to view the decrypted message.

One important use for public-key cryptography is to create online digital signatures. Digital signatures are used much like real signatures to verify who sent a message. The private key is used to sign the message, and the public key is used to verify the signature [1].

3.3 Hashing

Hashing is the last cryptologic algorithm and is a one-way function. Hashing turns any amount of data into a fixed-lengthed "fingerprints" that cannot be reversed [5]. This is great and is most commonly used for password protection. We are able to store the encrypted string into a database. This way the password plaintext is never stored on any hard drive, only the encoded string. When a user logs in, the entered password will be hashed by a function and this hash will be compared to the hash that is stored in the database. If the hashes match, the user is able log in.

```
hash("hello") = 2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7
hash("hbllo") = 58756879c05c68dfac9866712fad6a93f8146f337a69
hash("waltz") = c0e81794384491161f1777c232bc6bd9ec38f616560b
```

Figure 3: Strings being hashed [5]

In the above figure, figure 3, we see strings being hashed using a hashing function and then seeing the resulting hash. Using strong hashing functions allow similar strings like "hello" and "hbllo" to result in unsimilar hashes. Remember, in comparison to the other two algorithms, hashed data cannot be reversed. This means that hashing would not work for the example used in the previous two algorithms. If Tom sent a hashed message to Joe, Joe would not have any way to decrypt that message. Hashed data is not meant to be decrypted, it only is meant to be compared to, like in the case of passwords and logging into a system.

Common hashing algorithms include the Secure Hash Algorithm family, which includes: SHA-224, SHA-256, SHA-384, and SHA-512. The older SHA-1 and MD5 algorithms are currently in wider use, but flaws in both have been identified, and both should be retired in favor of a more secure hash [1].

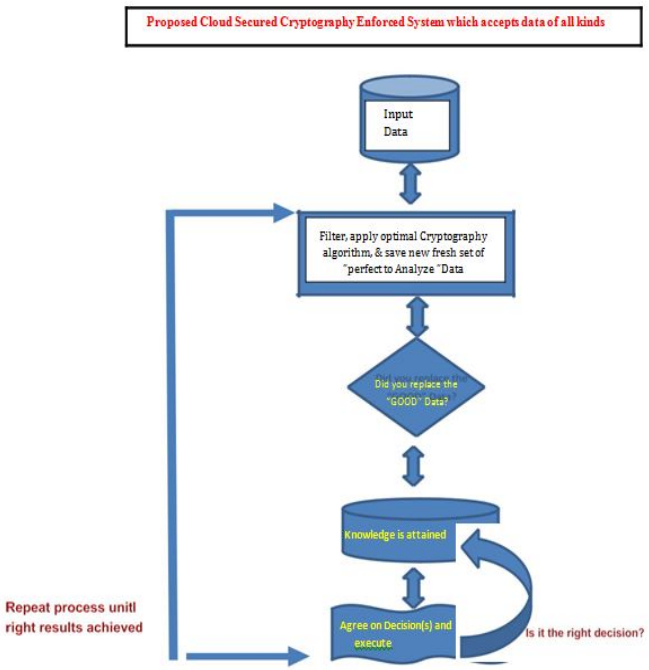
The symmetric and public key system in applying the concept of cryptology is such a great and normal proecdure in protecting user's data while the users use different programs that it carried over in principle and pratice over to protecting larger instances

Citation: Ihssan Alkadi (2017) Using Cryptology in Enforcing Database Security Traveling Through Networks and Cloud Computing. J Comp Sci Appl Inform Technol. 2(3):1-4

of data while using larger or same scale programs particularly reading and storing data in massive databaeses. Even though the concept remains the same when dealing with larger sets of data or databases the cryptologist has to worry the most about the incredible amount of records that may be not cleansed i.e, duplicates, or invalid. Either way the job of the security person is to enforce saving all transacrions and hiding the data from any outsiders soying or scanning the session online. Once data is saved then the key can be dismanteled.

In DBMS, the cryptology system should be used inside of the program? To come up with a cryptology system that works inside DBMS requires so many experimentations since it is difficult to write one just to work on one type of database as you know databases vary in types and compatibility. It is very and should be so easy to prevent the DBAs from getting the information they should not see by implementing and applying both a potent combination of encryption and 3-factor authentication logon mechanism.

Proposed "Optimal Cloud Secured Cryptography Enforced System" (OCSCES)



4 Cryptographic Risks/Attacks

Some of the issues we have solved from a system security standpoint by implementing a cryptological system are securing the sensitive data. Cryptology is best used for protecting confidential data in a system, i.e. passwords, signatures, high-importance messages. Cryptology changes the need of having to protect the entire system of data, to only have to protect one piece, the key. As long as the key remains secure, the encrypted data is meaningless to a perpetrator. This raises the question and biggest risk to using a cryptographic system: What happens if the key is modified, stolen, or lost?

4.1 Risks

If the key is lost, stolen, or modified you are left with a database full of meaningless data, there is no "restore" button or any reversal software. Still, with proper implementation and security, a cryptographic system is highly advantageous. It is a lot easier to focus your resources on protecting only one piece of data, the key, as opposed to protecting thousands of tables of data. The key must be protected with very strong access controls, and those controls must cover both direct and indirect access [1].

Direct access is access to the key itself. An attacker with direct access may copy the key and use it without fear of detection. Indirect access is access to an application or service that has access to the key. With indirect access, an attacker can feed encrypted data to the application or service and view the decrypted information [1].

4.2 Attacks

An attack on a cryptographic system is usually defined as a perpetrators attempt to retrieve the key or plaintext. A known-ciphertext attack is what most people think of a cryptographic attack [1]. This is where the attacker has access to all of the ciphertexts produced by a certain key. He then usually uses tools or other software to "guess" the plaintext. Dictionary, brute-force, and look-up tables are all common attacks used once an attacker has access to ciphertext.

A dictionary attack is an attack from a file containing words, phrases, and common passwords. Each word in the file is hashed, and is compared to the ciphertext that the attacker has acquired. These dictionary files are often filled with passwords from real databases, as well as "leet speak" equivalents. "Leet speak" is where "hello" would become "h3110" [5]. Brute-force attacks are much like dictionary attacks but instead of reading through a file, a brute-force attack will use all possible combinations of characters up to a given length. Brute-force attacks are slower and computationally expensive. Look-up tables are a pre-stored set of already hashed passwords, which are compared to a list of ciphertexts, which then return all the hashes that match, thus giving you the password.

Figure 4: Results of look-up table search [5]

5 Conclusion

There will always be new tools and methods for hackers to penetrate the latest security systems, just like there will always be bigger and better guns to penetrate the biggest and best armor. A well implemented cryptosystem is secure to some of the highest standards and allows any exposed data to be meaningless to the viewer, whether it be an attacker or a development member with read-only permissions. Using this type of system allows the secu-

Citation: Ihssan Alkadi (2017) Using Cryptology in Enforcing Database Security Traveling Through Networks and Cloud Computing. J Comp Sci Appl Inform Technol. 2(3):1-4

rity team to focus on keeping only one piece of data secure, the key, as opposed to thousands of sensitive data fields. Although losing, modifying, or having your key corrupted would be detrimental to a database system, with proper key management this can still be the best security practice available. It is important to implement this type of system correctly. Using a weak cryptosystem that is easy to crack or does not have proper key management can instill a sense of security confidence that renders your system less secure than it would have been had you not used cryptography at all.

6 References

1. Kevin K. Chapter 2: Securing Databases with Cryptography. Cryptography in the Database: The Last Line of Defense. Addison Wesley;2006.

2. Maurer U, Zurich ETH. The Role of Crypography in Database Security. 2013.

3. Beal V. Symmetric-key Cryptography. Webopedia. 2013.

4. Encrypted. N.p., n.d. Web. 2013

5. Salted Password Hashing - Doing It Right. Secure Salted Password Hashing. 2013.

6. Xianhui C, Lewis D. IPv6: Current Deployment and Migration Statu. IJRRCS. 2010;1(2)22-29.

7. Hura G, Singhal M. Data and computer communications: networking and internetworking. Boca Raton, FL: CRC Press. 2001.

8. Karrenberg D. The Internet Domain Name system explained for non-experts. Internet Society. 2004.

9. Morley D, Parker C. Understanding computers: Today and tomorrow, Comprehensive. 13th ed.Florence:KY: Course Technology; 2010.

10. Sosinsky B. Networking bible. Indianapolis:IN: Wiley Publishing;2009.

11. Tanenbaum AS. Computer networks. 4th ed.USA:Prentice Hall;2003.

12. Douglas M. Wireless security: is your data at risks? 2004.

13. Edge IE. Employ five fundamental principles to produce a SOLID, secure network. Information Security Journal: A Global Perspective. 2010;19(3):153-159. doi: 10.1080/19393551003649008

14. Goldsborough R. Technology Today: Computer zombies are out to get your machine. 2009. Available from: http://ccweek.com/article-1058-technology-today:-computer-zombies-are-out-to-get-your-machine.html

15. Reason J. Human Error. Cambridge: Cambridge University Press;1990.

16. Santaniello S. Don't forget acronyms. Teaching Pre K-8.. 2005;35(8):56.

17. Tanenbaum AS. Computer networks. 4th ed.Upper Saddle River:NJ: Prentice Hall;2003.

Citation: Ihssan Alkadi (2017) Using Cryptology in Enforcing Database Security Traveling Through Networks and Cloud Computing. J Comp Sci Appl Inform Technol. 2(3):1-4