ISSN: 2754-6705

Journal of Mathematical & Computer Applications

SCIENTIFIC Research and Community

Review Article

Open Access

Implementing Object Encryption in Software Applications: Methods and Insights

Krishna Mohan Pitchikala

USA

ABSTRACT

Everyone agrees that keeping information safe is crucial for software. When laws like HIPAA or GDPR require data to be encrypted, it's easy to find and follow the rules online. But when we are working on a small application and need to encrypt data, there's so much information available on data encryption that it can be overwhelming to choose the right approach. These studies cover a wide range of topics, from different ways to implement encryption to the most effective algorithms, often going beyond of what most applications need. The problem is that these high-level discussions can be intimidating for someone building their first software application. They might end up adding too much security or not enough. This paper aims to bridge that gap by discussing the different encryption methods and minimum encryption standards needed to protect information and showing how to achieve these standards. We will explore simple yet effective encryption techniques and practical steps to implement them in your application. By focusing on the basics, we hope to make the concept of data encryption accessible and manageable for beginners, ensuring they can protect their information without unnecessary complexity.

***Corresponding author**

Krishna Mohan Pitchikala, USA.

Received: September 06, 2023; **Accepted:** September 13, 2023, **Published:** September 20, 2023

What is Encryption?

Encryption is a process of converting human readable information into a text that looks like random gibberish. The purpose of doing this is to protect the sensitive information so that even if a hacker or any other attacker steals the data, they can't read it or misuse it because it's in a scrambled, unreadable format. A key is used in combination with an encryption algorithm to convert the data from readable to unreadable format. In the context of encryption, the human readable data is called plain text, and the unreadable data or scrambled data is called cipher text. This key when used to change plain text into cipher text is called encryption key and when it is used in the process of converting cipher text to plain text it is called decryption key.

Let's look at an example to see how encryption works in a simple way. Imagine Alice wants to send a secret message to Bob, but she doesn't want anyone else to be able to read it. To keep the message safe, she decides to use a method that scrambles the message so that it looks like nonsense to anyone who might intercept it.

Suppose Alice wants to send the message "Birthday Surprise" to Bob. Instead of sending it as it is, she decides to change each letter in the message by shifting it one position to the right in the alphabet. For example, 'B' becomes 'C,' 'i' becomes 'j,' and so on. After doing this, the message "Birthday Surprise" turns into "Cjsuiebz Tvsqsjtf."

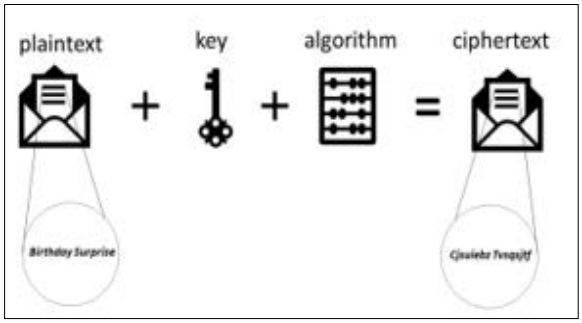

Figure1: Converting Plain Text to Cipher Text [1]

Now, Alice sends this scrambled message to Bob. When Bob receives it, he can't understand it right away because it looks like a bunch of random letters. But Alice also tells Bob the secret rule she used: each letter in the scrambled message needs to be shifted one position to the left to reveal the original message. When Bob follows this rule, he shifts the letters back and uncovers the message "Birthday Surprise."

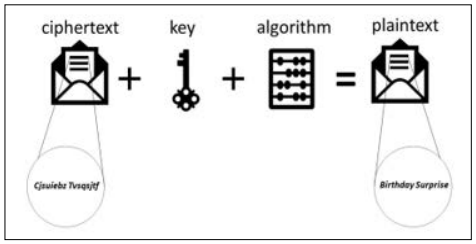

Figure 2: Converting Cipher Text to Plain Text [1]

In this Example

- **The key** is the rule about shifting letters by one position.
- **The encryption algorithm** is the method of applying this rule to the message.

This is how Alice and Bob can communicate without anyone else understanding their conversation. The message stays hidden because the key (the shifting rule) and the algorithm (the method of shifting) are known only to them [1].

Why is Encryption Important?

Encryption is a crucial step in protecting consumer data or application information from unauthorized access. By adding the extra layer of protection, encryption makes it much harder for security breaches to result in serious data leaks, keeping sensitive information safe and secure. Sometimes, access control mechanisms might be weak or compromised, allowing outsiders to gain access to sensitive information within an application. In such situations, encryption provides an extra layer of defense.

For example, imagine an application with two user roles: "read-only" and "admin." If certain data is encrypted and can only be decrypted by someone with admin rights, then even if a malicious person gains read-only access, they won't be able to do much harm. They might know the data is there, but they can't read or use it because it's still encrypted. This way, encryption ensures that even if unauthorized individuals gain some level of access, they are still unable to misuse or steal the information.

Types of Encryptions

There are two main types of encryptions: Symmetric and Asymmetric. They use different methods to encrypt and decrypt data. Each type has popular algorithms that are commonly used to secure information.

Symmetric Encryption

In symmetric encryption, the same key is used for both encrypting and decrypting the data. This means that both the sender and the receiver need to have the same secret key. Because this key is shared between them, symmetric encryption is also known as a shared key encryption.

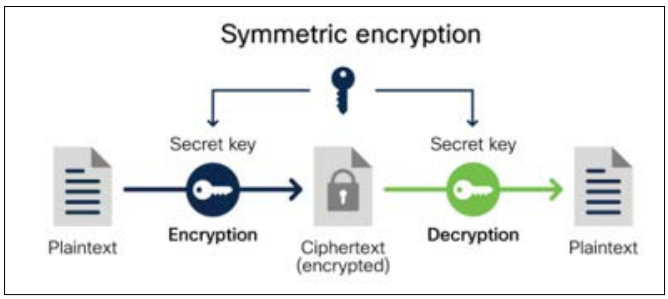

Figure 3: Symmetric Encryption [2].

Popular symmetric encryption algorithms include:

- Advanced Encryption Standard (AES)
- Triple Data Encryption Standard (Triple DES)
- Data Encryption Standard (DES)
- International Data Encryption Algorithm (IDEA)

Among all the encryption algorithms, AES is the well-known and effective methods of symmetric encryption used today. It uses block ciphers of 128, 192, or 256 bits to encrypt and decrypt data. AES is extremely secure it would take billions of years to crack so it's widely used to protect sensitive information in industries like government, healthcare, and banking. It's more secure than older methods like DES, Triple DES, and IDEA.

DES is now considered outdated by the National Institute of Standards and Technology (NIST) because it can no longer effectively protect data from brute-force attacks. NIST has completely withdrawn it as a standard, and even Triple DES, a more secure version of DES, is being discontinued due to growing security concerns. IDEA was created as a replacement for DES in the 1990s, but AES eventually proved to be more secure. While IDEA is still available as a free encryption method, it's largely considered obsolete and ineffective for securing sensitive information.

Symmetric encryption is also used in Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL). When a client connects to a server, they generate unique symmetric keys, known as session keys, to encrypt and decrypt the data shared during that session. Each new session between a client and server creates new, unique session keys. TLS/SSL uses both symmetric and asymmetric encryption to ensure that these client-server sessions and the information exchanged within them are secure.

Symmetric encryption is popular today because it's both secure and fast, and it's easy to use. Algorithms like AES are among the most secure encryption methods available in the market today, taking billions of years to crack through brute-force attacks. They are also very efficient, able to quickly encrypt and decrypt large amounts of data. Because of their strong security and speed, symmetric encryption methods like AES have become the gold standard and are widely used in many industries.

Even though the AES algorithm is fast and secure, symmetric encryption isn't the perfect solution for every application because the secret key must be shared between the sender and receiver. Symmetric encryption is best used in cases where a secure key exchange can be guaranteed, such as in private networks or for encrypted file storage [3].

Asymmetric Encryption

In Asymmetric encryption, a public key is used for encrypting the data and a private key is used for decrypting the data. Since it uses public keys for encryption, it is also called as public-key encryption. The public key and private key are mathematically dependent.

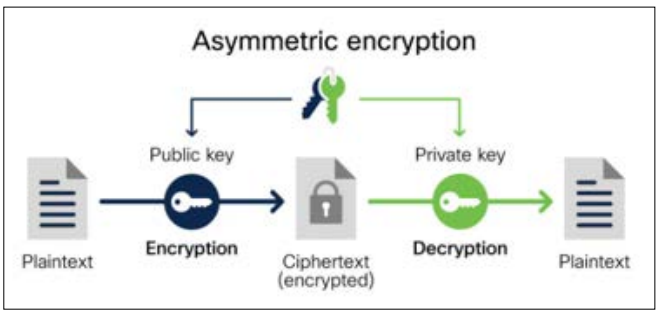

Figure 4: Asymmetric Encryption [2].

Asymmetric encryption solves the main problem of symmetric encryption by eliminating the need to securely share a secret key between the sender and receiver. Instead, the receiver creates a private key and shares a public key with everyone else.

Here's a Simple Example: Alice wants to receive messages from multiple people, but she wants to be the only one who can read them. If she used symmetric encryption, she would have to send the same secret key to everyone who wants to send her messages so they could encrypt them, and she could decrypt them. However, with asymmetric encryption, Alice creates a private key that she keeps secret and shares a public key with everyone. Anyone can use Alice's public key to encrypt a message for her, but only Alice can decrypt it using her private key.

Asymmetric encryption also supports digital signatures. Alice can sign data with her private key, and anyone with her public key can verify that she was the one who signed it and that the data hasn't been changed. This concept was once thought impossible until the 1970s, but it's now widely used, such as in web browsers.

Popular Asymmetric Encryption Algorithms Include

- Rivest Shamir Adleman (RSA)
- Digital Signature Standard (DSS)
- Elliptical Curve Cryptography (ECC)
- Diffie-Hellman exchange method

RSA is the widely used asymmetric encryption algorithm. RSA works by generating a public key through the multiplication of two large, random prime numbers, and a private key is also generated using these same numbers. Once these keys are created, information can be encrypted with the public key and decrypted with the private key.

In the DSS, which includes the Digital Signature Algorithm (DSA), a sender uses their private key to sign a message or file, and the recipient uses the sender's public key to verify that the signature is authentic and came from the correct source. ECC is an alternative to RSA that uses smaller key sizes and mathematical curves to perform asymmetric encryption. ECC is faster than RSA in generating keys and signatures, and it is considered the next big thing in asymmetric encryption, especially for securing web traffic and cryptocurrency. Diffie-Hellman is another breakthrough in cryptography, offering a way for two parties who have never met to securely exchange public and private keys over public, insecure communication channels. Before Diffie-Hellman, parties had to physically exchange encryption keys in advance to communicate securely.

With asymmetric encryption, private keys stay secure and are never shared over unsafe channels. This provides extra protection because the keys needed to decrypt sensitive information are never exposed, reducing the risk of being compromised. It also allows senders to use their private keys to sign messages or files, so the recipient can verify that the message really came from the sender and not someone else.

While asymmetric encryption provides strong security, it's slower than symmetric encryption. This is because it uses longer key lengths and more complex calculations. The public and private keys in asymmetric encryption are mathematically linked, so theoretically, a public key could be used to crack a private key. To prevent this, asymmetric encryption uses very long key lengths, making it almost impossible to break with current technology [3].

Techniques to Secure Data

Now that we know what encryption is, why it's important, and the different types and algorithms used to encrypt data, let's discuss how to use encryption in software applications. In any software application, data exists in one of three states:

1. **Data at Rest:** This refers to data that is stored somewhere, like on a hard drive, a server, or in a database. It's not actively being used or transferred; it's just sitting there, "at rest."
2. **Data in Transit:** This is data that is actively moving from one place to another, such as when you send an email, browse a website, or transfer files over the internet. It's "in transit" as it moves from one device or location to another.
3. **Data in Use:** This is data that is currently being accessed, processed, or used by a computer or application. For example, when you're editing a document or running a program, the data involved is "in use."

Securing data in use is most difficult because it needs to be accessible for legitimate use while being protected. This is especially important when sensitive information is being processed. Traditional encryption methods safeguard data when it's stored or transmitted, but not when it's actively being used. Although new techniques are being developed to handle this issue, we won't cover those in this paper since we're focusing on minimum encryption standards.

Data in transit can be intercepted by attackers if not properly protected, so it's crucial to secure it. The most common and effective way to do this is by using Transport Layer Security (TLS). TLS is a security protocol that has replaced the older SSL protocol and is widely used across the internet to protect data as it travels between a client (like your web browser) and a server (like the website you're visiting). When you visit a website that starts with "HTTPS," you can feel confident that TLS is being used to secure the connection. This means that any data you send or receive, like passwords or personal information, is encrypted and is safe. So, in any software application that involves communication with users or clients, it's essential to make sure that it uses HTTPS. This involves configuring the client software to support TLS, ensuring that data is securely transmitted, and the users' information is protected.

Similarly, data at rest data that is stored on a server, in a database is also at risk if it's not properly secured. To prevent unauthorized access to stored data, it's crucial to encrypt it. One can use publicly available services to encrypt the data like Amazon Key Management Service (Amazon KMS) which takes care of regularly rotating and updating the keys to protect against evolving threats.

Server-Side Encryption vs Client-Side Encryption

Two common approaches for implementing encryption in software applications are server-side encryption and client-side encryption. Understanding the differences between these methods and knowing when to use each is essential to make informed decisions about securing the data.

Server-Side Encryption

Server-side encryption is when data is encrypted and decrypted by the server where it's stored or processed. When the data is sent to the server, it encrypts the data before saving it and decrypts it when needed. The server also takes care of managing the encryption keys, often using built-in services from cloud providers like AWS, Google Cloud, or Azure, or through custom encryption solutions. Since the server handles all the encryption and decryption, the user doesn't need to worry about these steps, making it easier for them. Server-side encryption is commonly used in cloud storage services, like Amazon S3, which automatically encrypts the data stored there. It's also used in databases to protect information,

even if the database is compromised. This method is popular for web applications where the server is trusted and secure.

Client-Side Encryption

Client-side encryption is when data is encrypted and decrypted on the user's device before it is sent to the server. This means that the server only stores encrypted data, and when the user retrieves it, the data remains encrypted until it's decrypted on the user's device. In this setup, the user is responsible for managing the encryption keys, and the server cannot access these keys, making it impossible for the server to decrypt the data. This approach offers strong security benefits because even if the server is compromised, the attacker can only access encrypted data, which is useless without the decryption key. Client-side encryption is commonly used in end-to-end encrypted messaging apps like WhatsApp and Signal, where only the communicating parties can read the messages, not even the service provider. It's also ideal for protecting highly sensitive data, such as in healthcare or financial applications, and in situations where one can't fully trust the server or want to protect against insider threats.

Comparing Server-Side and Client-Side Encryption

	Server-Side Encryption	Client-Side Encryption
Who Encrypts the Data?	The server encrypts and decrypts the data	The client encrypts data the before sending it to the server
Security	Server has access to plaintext data	Server only sees encrypted data
Complexity	Easier for clients; server handles encryption	More complex for clients; as they handle encryption.
Use Cases	Cloud storage, web applications, databases	End-to-end encrypted messaging, sensitive data protection, zero trust scenarios

Conclusion

Encryption is an essential tool that every software application should use to protect data. When selecting an encryption algorithm, it's important to consider your specific security needs, performance goals, and how well the algorithm meets industry standards, along with its reliability and suitability for your application. Even for simple applications, it's crucial to encrypt data both at rest and during transmission. The choice between server-side and client-side encryption depends on your security needs: use server-side encryption for a simpler, centralized approach where the server manages everything, especially if you trust the server's security. Opt for client-side encryption when you need the highest level of data confidentiality or when users need full control over their encryption keys. In some cases, a mix of both methods might be the best approach, depending on the type of data and its security requirements. By understanding how to protect data in different states and the various algorithms available, you can build secure applications.

References

1. Daniel Adetunji (2023) Symmetric and Asymmetric Key Encryption Explained in Plain English https://www.freecodecamp.org/news/encryption-explained-in-plain-english/
2. See, Try, or Buy a Cisco Secure Firewall https://www.cisco.com/c/en/us/products/security/encryption-explained.html
3. (2021) Symmetric vs. Asymmetric Encryption: What's the Difference? https://www.trentonsystems.com/en-us/resource-hub/blog/symmetric-vs-asymmetric-encryption
4. (2019) What Are the Different Types of Encryption? https://www.hp.com/us-en/shop/tech-takes/what-are-different-types-of-encryption
5. https://cyscale.com/blog/types-of-encryption/
6. Types of Encryption - Everything You Need to Know for 2022 (2022) https://inspiredelearning.com/blog/types-of-encryption-everything-you-need-to-know-for-2022/
7. What is a cryptographic key? https://www.cloudflare.com/learning/ssl/what-is-a-cryptographic-key/
8. Dominic Fraser (2018) What are encryption keys and how do they work? https://medium.com/codeclan/what-are-encryption-keys-and-how-do-they-work-cc48c3053bd6
9. https://www.linkedin.com/pulse/white-paper-data-security-encryption-secure-/
10. https://accelerationeconomy.com/cybersecurity/how-to-secure-the-3-states-of-data-at-rest-in-motion-in-use/
11. Margaret Rouse (2017) What Does Encryption Key Mean? https://www.techopedia.com/definition/25403/encryption-key
12. Data Encryption Algorithm Methods & Techniques (2023) https://www.digitalguardian.com/blog/guide-data-encryption-algorithm-methods-techniques
13. Arnaud (2023) Symmetric vs Asymmetric Encryption: What's the difference? https://blog.mailfence.com/symmetric-vs-asymmetric-encryption/
14. Franklin Okeke (2022) Asymmetric vs symmetric encryption: What's the difference? (2022) https://www.techrepublic.com/article/asymmetric-vs-symmetric-encryption/