

Cryptography: The Hidden Key to Secure Communications

Wayne Ricks Jr
CYSE 463
Cryptography in Cybersecurity
Professor Susan Zehra
April 28, 2025

Abstract

In the field of Cybersecurity, cryptography is essential for all forms of secure communication. This is a practice that has been used for centuries by leaders and scholars to share information secretly with allies or to protect the integrity of the information being transferred. Cryptography is a tool that utilizes encryption techniques to encode sensitive information that must not be publicly known. The evolution from simple ciphers to sophisticated infrastructures used today highlights the resilience that comes from perfecting a practice such as this. With breakthroughs such as the Diffie-Hellman Exchange back in 1976, this helped in modernizing cryptography, strengthening its security and overall complexity. These techniques have been seamlessly integrated into each of our daily lives through digital experiences. With applications such as mobile banking and the overall need to protect personal data, these techniques ensure that we can rely on our systems to remain resilient and trustworthy against cyber attacks. Modern cryptography can best be broken down into 3 main categories: Symmetrical, Asymmetric, and Hash Functions. Each of these domains encase the same idea of security, where a key is used to encrypt and decrypt the data, but each domain uses a different approach to achieve its purpose. These keys can be generated, exchanged, stored, and ultimately retired within systems that ensure confidentiality, integrity, and accessibility as it traverses across various networks. Together, these cryptographic elements form the backbone of digital security. Symmetric encryption offers speed, asymmetric encryption provides scalability and secure communication, and hash functions ensure integrity and verification. Their combined application secures modern systems against threats, fostering trust in an increasingly digitalized world.

Symmetric Key Algorithms

One of the most widely used forms of cryptography are Symmetric Key Algorithms. What makes this algorithm so popular among users, in comparison to asymmetric algorithms or hash functions, is the usage of only a single key to encrypt and decrypt the information. Having only a single key to do both jobs makes its computation less intensive and allows large amounts of data to be processed at a higher efficiency. In environments where processing power is limited such as mobile devices, it is extremely beneficial to use this algorithm for sending information across multiple networks without it being intercepted. The oldest known symmetric key algorithm can be dated back to the Roman Empire, where ruler Julius Caesar used an algorithm that took letters and scrambled them a set number of positions, where the key will explain how many shifts must take place in order to properly decrypt the ciphertext. Fast-forward to the 20th century and innovations within symmetric key algorithms began to lay the groundwork for the algorithms used today with the help of brilliant pioneers of cryptography such as Whitfield Diffie, Martin Hellman, and Ralph Merkle.

Symmetric key algorithms can be grouped into two categories, which are based on how they process data: Block ciphers and Stream ciphers. In Block Ciphers, data is processed in fixed-sized blocks ranging from 128-bits to 256-bits at a time. Some examples of block cipher algorithms include the Data Encryption Standard (DES), Triple DES (3DES), and the Advanced Encryption Standard (AES). In this algorithm, there are various ways of operation in order to encrypt and decrypt the data. Electronic Codebook (ECB) mode takes each block independently and encrypts the plaintext. This mode is rarely used due to the nature of its encryption. Any plaintext that looks identical to each other will produce identical ciphertexts. Cipher Block Chaining (CBC) mode takes the ciphertext produced from the previous block and XORs it with the plaintext of the next block of data. XOR is the process where two inputs are combined to produce a single output. In order to start this method with the first block, an Initialization Vector (IV) is required for proper computation. Counter (CTR) mode takes a block of data and converts it into a stream cipher by combining the plaintext with a counter value. This mode is widely used because it enables parallel processing and computation for both encryption and decryption of data. Galois/Counter (GCM) mode utilizes a universal hashing over a binary field to provide not only encryption, but also authentication to prove that the encrypting party was indeed the one who encrypted the data. This method is also widely used due to its integrity for data and commonly used for secure communications. Unlike block ciphers, where data is grouped together and encrypted, Stream ciphers encrypt data bit by bit as it is being transmitted. Stream ciphers are widely utilized in applications like video calls or video streaming. Another noticeable difference with Stream ciphers is the variability that comes with using this algorithm. Block

ciphers require data to be a fixed-length and if the data doesn't meet the requirements, then padding is added to compensate. Stream ciphers have the ability to handle variable-length data streams without the need for padding.

Asymmetric Key Algorithms

This form of cryptography, also known as Public-Key cryptography, is a technique that utilizes a pair of keys to ensure the integrity of secure communications. When it comes to the keys, they are mathematically related, but are used to serve different purposes. The public key is the key that is seen by anyone and can be used for encryption, where the private key is the key that is only seen by the owner of it and is used for decryption. The generation of the keys consists of mathematical computations using large prime number factorization and the selection of computation method, whether it be one number for the elliptic curves or two numbers for the RSA modulus. These computations are meant to be easy to perform in one direction but extremely difficult to bypass in the opposite direction without the proper specific information. When engaging with asymmetric encryptions, it is imperative that one understands the functionality of digital signatures. This cryptographic mechanism is used to verify the integrity and authenticity of any digital message or document. If Person A wants to send Person C a message without Person B knowing the contents, then Person A would sign the message with their private key, creating a cryptographic hash combined with their key. Once the recipient, Person C, receives the message, they are then able to take Person A's public key in order to ensure that the message indeed came from Person A and to verify the message was not altered or manipulated during its transmission.

Unlike symmetric encryption, where algorithms can be classified into groups based on their means of processing data, asymmetric encryption algorithms each have different properties and applications that aid in securing communications. One of the most popular asymmetric algorithms is the Rivest-Shamir-Adleman algorithm, better known as the RSA algorithm. Created in 1978, this algorithm utilizes the factorization of large prime numbers in order to protect data contents. When it comes to deciding what prime number is suitable for security, the larger the prime number is, the greater the security provided after its computation. This algorithm gained its popularity due to reliability, despite being used for decades. Major security protocols like HTTPS, SSL, and email encryption rely on the functionality of RSA in order to properly complete their tasks. Another widely-used asymmetric algorithm is the Elliptic Curve Cryptography (ECC). This algorithm, as stated in the name, utilizes integers on an elliptical curve that are scalar. The ECC algorithm, compared to traditional algorithms like RSA, gained its popularity particularly due to its efficiency. ECC generated keys are much smaller than the keys generated by RSA, while having the same strength of security. If an ECC key was to be 256-bits, this holds the same strength of security as a 3,072-bit RSA key. With smaller key sizes, this results in faster computations, reduced storage requirements, and less bandwidth usage which makes this algorithm more preferable for IoT devices that have limited computational

power such as mobile devices and digital certificates. Asymmetric encryption, despite its intensive computational intensity, is vital for the protection of data in an environment of unprecedented connectivity.

Cryptographic Hash Functions

Hashing, unlike encryption, is a technique that ensures data integrity by converting input data into a fixed-length string, also known as a hash or digest. When compared to other forms of encryption where there is a 2-way process to send and receive the data, hashing is a one-way process that creates unique strings for each different input entered. Some notable properties of a good hashing function include features like high resistance to collisions, deterministic outputs, and being computationally infeasible to reverse. Collision resistance refers to a system being able to deny the entry of two inputs that could generate the same output. Deterministic outputs means that every output that is generated will be unique to the input that was entered. The generated outputs must also be complex enough where nothing will be able to undo once hashed, no matter the amount of computations. That the original input can not be reconstructed directly from the hash. Many of the applications we used on a daily basis utilize hashing in various ways from password storage and Message Authentication Codes (MACs), to digital signatures and data integrity verifications. In password storage hashing, it is good practice to do what's called "salting" and "peppering" before hashing. This practice refers to adding a unique and random value to the password to differentiate users in the case their passwords are similar. Peppering is done in a similar fashion to salting where their difference lies in the random value added. This value is kept a secret and is not stored with the password, adding an additional layer of security. Hashing functions have been popularized over the decades due to their resilience and development throughout the years at overcoming vulnerabilities.

A popular hash function that was used frequently was the Message Digest Algorithm 5, or MD5. Created in 1992 by Dr. Ron Rivest, this algorithm was designed originally for applications with digital signatures, it quickly widespread to other applications and was used in a variety of applications. This algorithm would generate a 128-bit hash value that was represented as 32-character hexadecimal number. Despite its popularity, the algorithm became obsolete due to high collision vulnerabilities, making it susceptible to the compromising of systems. Another popular hash function is the Secure Hash Algorithm, or SHA. This function is the perfect example of what was aforementioned such that over time, these functions will need to be developed to combat against attackers. The first version, commonly known as SHA-1, laid the foundation for the other functions to flourish once it was depreciated. This function's duty consisted of producing 160-bit hash values, but due to advancements in computer technology and the constant evolution of cybersecurity vulnerabilities, the function became a liability, as it was very prone to collision attacks. The successor to this function is the SHA-2 function, which is made up of a family of functions. Some variations you may see in this family includes SHA-224,

SHA-256, SHA-384, and SHA-512. This group capitalized on the shortcomings of the SHA-1 function, providing enhanced security using mathematical constructs. The SHA-3 function is the latest development surrounding these functions, being only a decade in age. Unlike its predecessors, which were based on the Merkle-Damgård construction, this function was developed around a construct that's known as the Sponge construct. This design provided more security against attacks such as collisions, pre-images, and length extensions.

References

1. National Institute of Standards and Technology (NIST). (2015). [Cryptography | NIST](#)
2. Damgård, I. (1989). A Design Principle for Hash Functions. Advances in Cryptology — CRYPTO 89 Proceedings, 416–427. One Way Hash Functions and DES | SpringerLink
3. - Diffie, W., & Hellman, M. (1976). New Directions in Cryptography. IEEE Transactions on Information Theory, 22(6), 644–654.
4. - Rivest, R. L., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, 21(2), 120–126.