# Introduction to Information Security

Christian J. Rudder

November 2024

Contents

*This page is left intentionally blank.*

Big thanks to **Professor Sharon Goldberg**
for teaching CS357 (Introduction to Information Security)
at Boston University.

*The following five sections are pre-requisites from Concise Work Section Modules [21].*
**Available at:** https://github.com/Concise-Works/sect-modules.
These are not needed, but are recommended for a better understanding of the material.

— 1 —

Software Security & Defenses

## 1.1 Document Certificates & Binding Encryptions

Before, all communication sent "**over the wire**" (from device to device), was sent "**in the clear**" (unencrypted). This means that anyone could view data sent between devices in plain text. This is a problem when setting up infrastructure such as banking, e-commerce, or any other service that requires sensitive information to be sent over the internet.

**Definition 1.1: Integrity & Authenticity**

**Integrity** is the assurance that data has not been altered in transit.
**Authenticity** is the assurance that the data is coming from the correct source.

**Definition 1.2: Transport Layer Security (TLS)**

TLS is a protocol providing end-to-end encryption of data. It authenticates the server via **TLS certificates** to ensure the client is connecting to the correct host. It also ensures integrity of the data.

The Engineering Task Force (IETF) published the first version of TLS in 1999. As of today the most recent version is TLS 1.3. (2018). [6]

**Definition 1.3: Secure Sockets Layer (SSL) [Deprecated**

SSL is the predecessor to TLS. It was developed by Netscape in the 1990s. SSL 3.0 was released in 1996. SSL 3.0 was found to be insecure and was replaced by TLS 1.0 in 1999. [6]

**Definition 1.4: Certificate Authority (CA)**

A CA is a third-party entity that issues digital certificates. Often called **SSL certificates** or TLS certificates. The protocol supports both SSL and TLS. Despite SSL's deprecation the name stuck due branding issues. Browsers and Operating systems have a list of trusted CAs called the **root store**. A full list of Microsoft's trusted CAs can be found here:
https://ccadb.my.salesforce-sites.com/microsoft/... [13]

**Definition 1.5: Encryption**

**Encryption** is the process of converting plaintext into ciphertext (indiscernible text).
**Decryption** is the process of converting ciphertext back into plaintext.

**Definition 1.6: Symmetric & Asymmetric Encryption**

**Key**: is a seed/piece of information used to encrypt or decrypt data.
**Symmetric Encryption**: uses the same key for both encryption and decryption.
**Asymmetric Encryption**: uses a public key for encryption and a private key for decryption. [2]

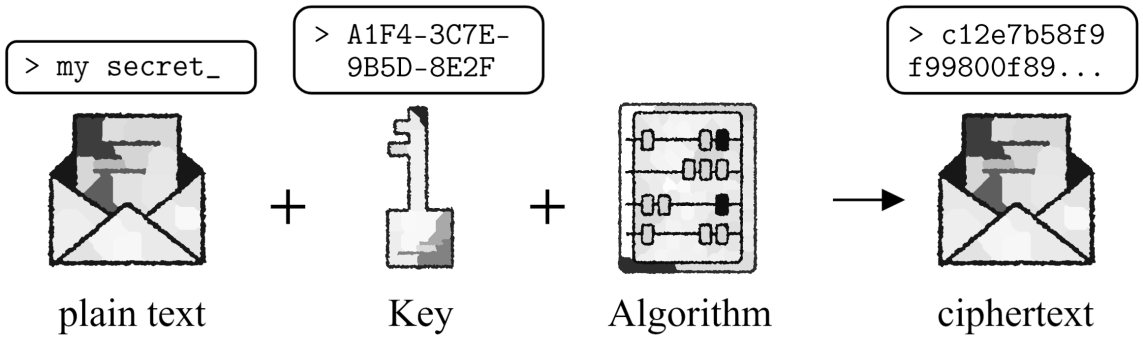



Figure 1.1: High-level depiction of encryption.

Encryption takes a key, data, and an algorithm to produce ciphertext. Decryption takes the same key, ciphertext, and algorithm to produce the original data.

**Definition 1.7: Hashing**

Hashing is the process of converting data into a fixed-length string of characters. Hashing is a one-way function, meaning it cannot be reversed (theoretically). In practice, it is computationally infeasible to reverse a hash without brute force (trying all possible inputs) or exploiting weaknesses in the hashing algorithm. Some hash algorithms use the text itself as input, while others may incorporate a separate key (e.g., HMAC). [8]

**Definition 1.8: Hypertext Transfer Protocol Secure (HTTPS)**

A version of HTTP that uses TLS to encrypt data. [7]

**Definition 1.9: SSL/TLS Certificate Specifications**

- **Common Name (CN)**: The domain name the certificate is issued for.
- **Subject Alternative Name (SAN)**: Additional domain names or subdomains covered by the certificate.
- **Key Length**: A minimum of 2048 bits, ensuring strong encryption.
- **Hashing Algorithm**: Typically SHA-256 for secure data integrity.
- **Valid From/To**: The validity period, usually up to 397 days.
- **Issuer**: The trusted Certificate Authority (CA) that issued the certificate.
- **Extended Key Usage**: Specifies purposes like server authentication or client authentication.

[13]

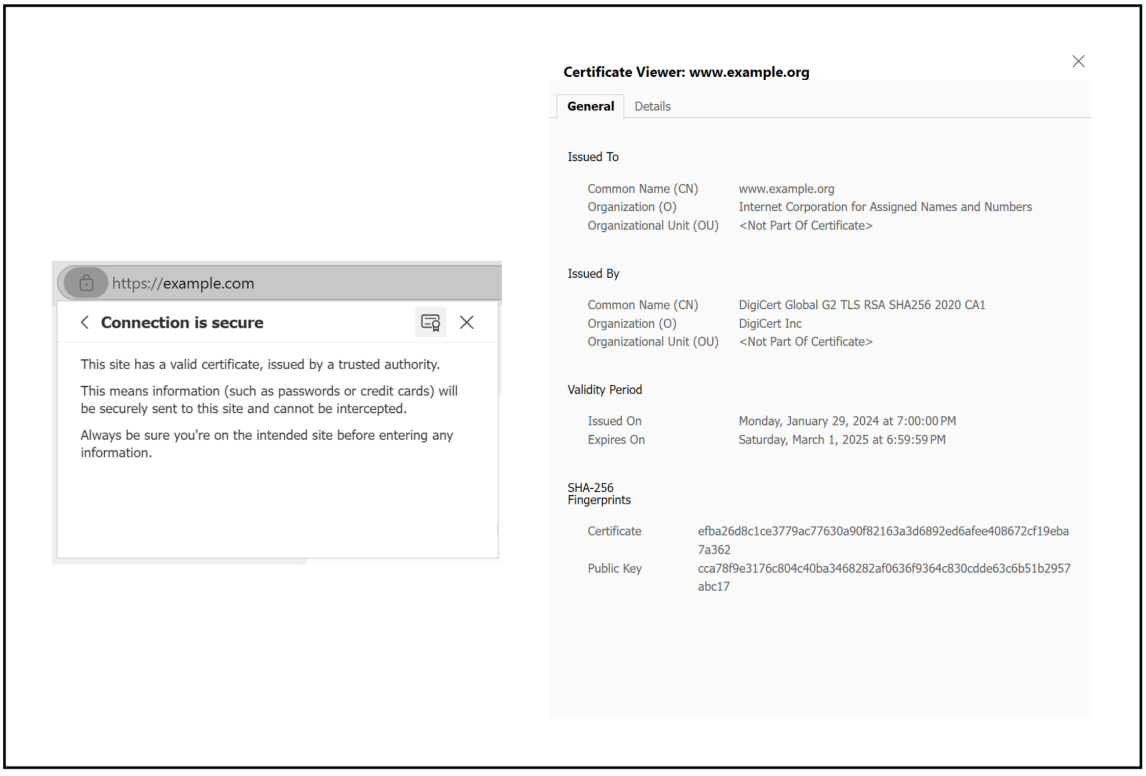



Figure 1.2: SSL certificate obtained through the Edge browser on example.com

**Definition 1.10: Public Key Infrastructure (PKI)**

PKI is a system for managing digital certificates. It includes the creation, distribution, and revocation of certificates. PKI is used to secure data transmission over the internet such as secure email, VPNs, and other services. [16]

**Definition 1.11: Digtial & Authority Certificates**

**Digital Certificate**: An electronic document binding and proving ownership of a public key. **Authority Certificate**: or **Root Certificate** issued by a CA, signs other certificates. it is **self-signed** and is the top of the certificate chain.

This establishes layers of trust and distance between the root certificate and the end-user certificates. If the root certificate is compromised, all certificates through the chain are compromised. [22]

**Definition 1.12: Digital Signatures**

To verify a source, a **digital signature** is used.

- **Key Generation**: A private and public key pair is generated beforehand.
- **Hashing**: A cryptographic hash of the data is created (e.g., SHA-256).
- **Encryption**: The private key encrypts the hash, creating the digital signature through an algorithm (e.g., RSA).
- **Verification**: The recipient decrypts the signature with the public key and compares the result to their own hash of the data. [10]

**Definition 1.13: Certificate Signing Request (CSR)**

To obtain a digital certificate, a client generates a CSR, which involves the following steps:

1. **Generate Key Pair**: Create a private key (kept secret) and a public key (shared).
2. **Produce CSR Data**: Include identifying information (e.g., Organization (O), Common Name (CN)), the public key, and other details in a standardized format such as X.509.
3. **Sign the CSR**: Hash the CSR data and sign it using the private key to prove ownership.
4. **Submit and Issue Certificate**: Submit the CSR to a CA, which validates the signature, signs the certificate with its own private key, and issues the certificate. [15]

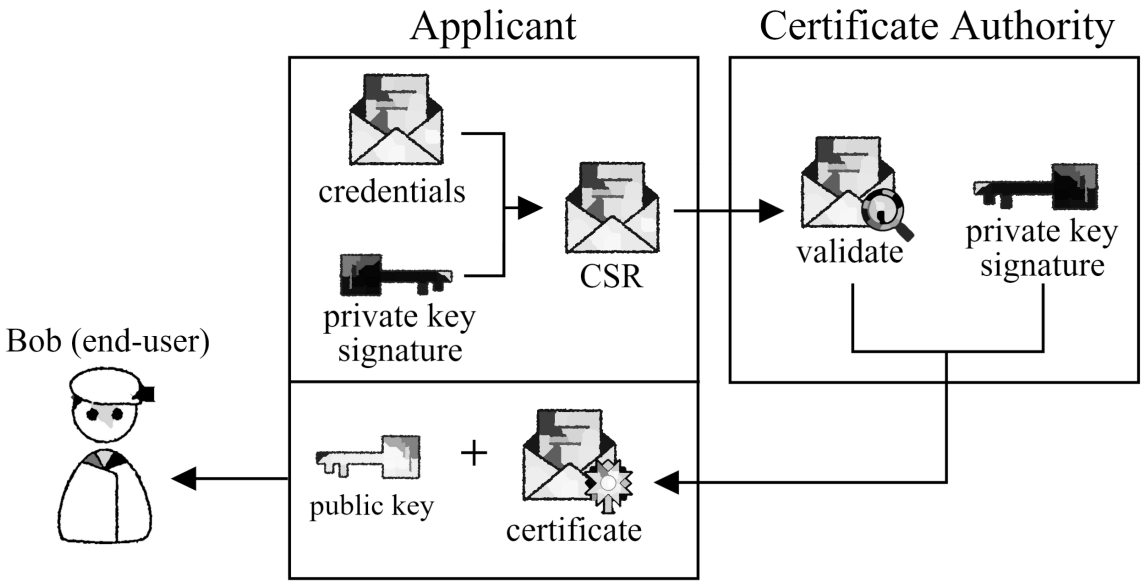



Figure 1.3: Example of a Certificate Signing Request (CSR)

**Definition 1.14: Certificate Revocation List (CRL)**

A CRL is a list of certificates that have been revoked by the CA before their expiration date. This is used to prevent the use of compromised certificates. [9]

**Definition 1.15: Chain of Trust**

A **Chain of Trust** is a hierarchical sequence of certificates used in PKI to establish trust between entities. It consists of:

- **Root Certificates**: Self-signed certificates at the top of the trust chain, trusted directly by operating systems and browsers.
- **Intermediate Certificates**: Issued by the root CA to delegate trust, adding a layer of security by isolating the root CA from direct interactions.
- **End-Entity Certificates**: Issued to users, servers, or devices to authenticate their identity and enable secure communications.

Each certificate is digitally signed by the private key of the certificate authority above it in the chain, with the root certificate serving as the ultimate trust anchor. Certified issuers are preferred over self-signed certificates because they undergo rigorous external validation, creating a verifiable path of trust. [9]

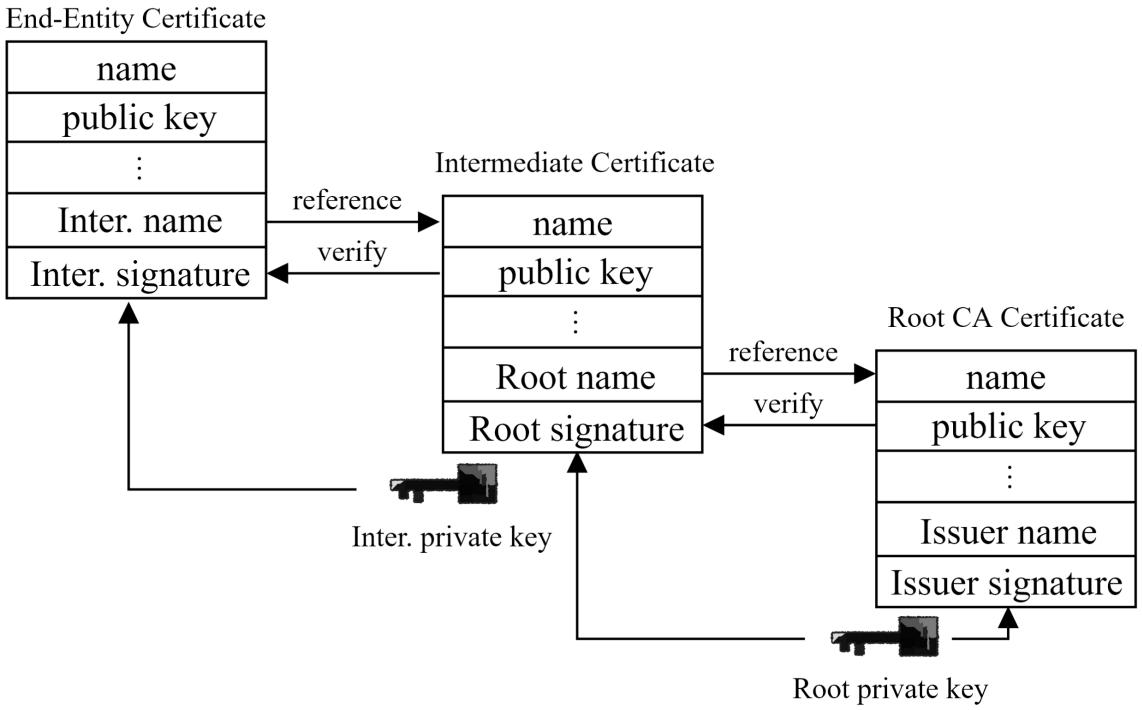



Figure 1.4: A Chain of Trust from the Root Certificate to the End-Entity Certificate.

**Definition 1.16: Root Certificate Authority (CA) Signing Ceremonies**

A **Root Certificate Authority (CA) Signing Ceremony** is a highly secure process in which a Root CA's private key is used to sign subordinate certificates, establishing trust within a Public Key Infrastructure (PKI). Key characteristics include:

- **Rigorous Security**: Conducted in offline, access-controlled environments with multiple layers of physical and procedural security. Entry requires the presence of multiple trusted individuals simultaneously.
- **Defined Roles**: Roles such as Crypto Officers, Witnesses, and Administrators are assigned to ensure transparency and accountability.
- **Global Trust Anchors**: Managed by organizations like ICANN for DNSSEC, these ceremonies protect the integrity of critical internet infrastructure.
- **Independent Operations**: Root CAs are typically independent from government oversight, though some, like the U.S. Department of Defense, manage government-affiliated Root CAs.

Learn More: https://cloudflare.com/learning/dns/dnssec/root-signing-ceremony/ [5]

**Definition 1.17: DNS over HTTPS (DoH) and DNS over TLS (DoT)**

**DNS over HTTPS (DoH)** and **DNS over TLS (DoT)** are protocols designed to encrypt DNS queries, improving privacy and security:

- **DoH**: Encrypts DNS queries over the HTTPS protocol (port 443), making them indistinguishable from regular HTTPS traffic.
- **DoT**: Encrypts DNS queries using the TLS protocol (port 853), ensuring DNS requests are secure and tamper-proof. Though because of its use of port 853, traffic is more easily identifiable as DNS, making DoH the preferred method.

Both protocols prevent DNS queries from being visible in plaintext, mitigating risks like eavesdropping and DNS spoofing. DNS security is known as **DNSSEC**. [4]

## 1.2 Encryption Algorithms & Security Definitions

In the previous section the notion of encryption was introduced (asymmetric and symmetric). Here adversaries (i.e., attackers, or hackers) will be defined along with the specifications an algorithm must meet.

**Definition 2.1: Adversaries**

Adversaries are entities that attempt to break the security of a system. There are two types of adversaries:

- **Eavesdroppers:** Can only intercept and read messages.
- **Man-in-the-Middle (MitM):** Can intercept, read, and modify messages.

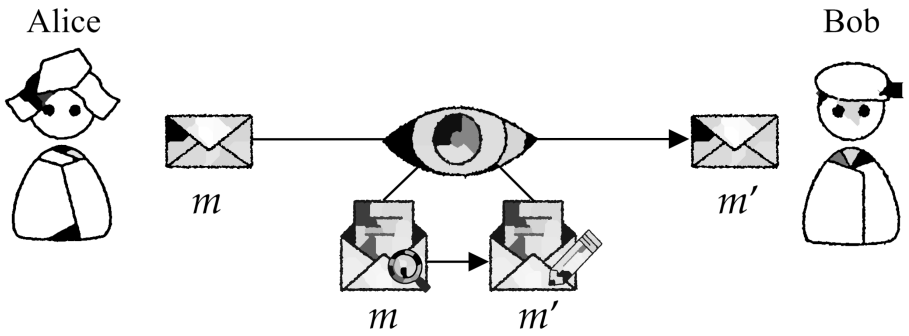



Figure 1.5: A MitM Attack, reading and altering the contents of $m$ and sending $m'$.

Instead plain variables $A$ and $B$, often Alice and Bob are used. There are other common for entities, learn more here: https://en.wikipedia.org/wiki/Alice_and_Bob.

**Definition 2.2: Security Definitions**

Security definitions formalize the properties a system must satisfy to resist adversarial attacks. These include:

- **Confidentiality:** Ensures that adversaries cannot learn the contents of the message.
- **Integrity:** Guarantees that adversaries cannot alter the message without detection.
- **Authenticity:** Verifies that the message originates from the claimed sender and has not been tampered with.

**Theorem 2.1: Kerckhoffs's Principle**

*"Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi."*

**Literal translation**: [The method] must not be required to be secret, and it must be able to fall into the enemy's hands without causing inconvenience [17].

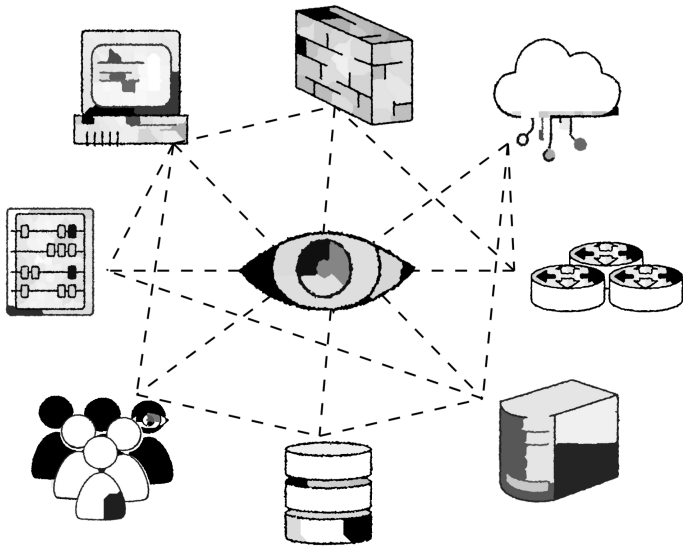


Figure 1.6: Kerckhoffs's Principle, an adversary's unbounded sight.

I.e., **The adversary knows all**—the algorithm, the architecture, an insider, any exploit—all communication is naked, visible on the wire. Though despite all, is secure. This is the essence of Kerckhoffs's Principle.

**Definition 2.3: Non-cryptographic Security**

A system which does not follow Kerkhoff's Principle: security through obscurity (hiding the algorithm) is not a cryptographic. As if any rosetta stone is found, the system is compromised.

**Note**: The rosetta stone was a slab of stone found in 1799, which helped decipher Egyptian hieroglyphics. I.e., a key between languages. Learn more: https://wikipedia.org/rosetta_stone.

The rest of the section will cover methods used over centuries to attempt to secure messages.

**Theorem 2.2: Caesar Cipher**

The Caesar Cipher is a non-cryptographic scheme, named after Julius Caesar, dating back 45BC to protect military communications. Each letter in the plaintext is shifted $x$ places down the alphabet. E.g., $x = 3$, 'A'='D', 'B'='E', and so on. [19]

**Note**: Here is a fun online tool to try the Caesar Cipher: https://cryptii.com/caesar-cipher.

**Theorem 2.3: Vigenère Cipher**

The Vigenère Cipher is a non-cryptographic scheme, created in mid-1500's by italian cytologist Giovan Battista Bellaso, later popularized and misattributed to Blaise de Vigenère. It addressed the Caesar Cipher's weakness by using attributing odd and even digit places to different Caesar Ciphers.

This was aimed to stop frequency analysis attacks, as for instance, 'E' is the most common letter in English. If a letter is repeated at high frequency, it is likely 'E'. [19]

| 0 | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ | $g$ | $h$ | $i$ | $j$ | $k$ | $l$ | $m$ | $n$ | $o$ | $p$ | $q$ | $r$ | $s$ | $t$ | $u$ | $v$ | $w$ | $x$ | $y$ | $z$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $f$ | $g$ | $h$ | $i$ | $j$ | $k$ | $l$ | $m$ | $n$ | $o$ | $p$ | $q$ | $r$ | $s$ | $t$ | $u$ | $v$ | $w$ | $x$ | $y$ | $z$ | $a$ | $b$ | $c$ | $d$ | $e$ |
| 2 | $k$ | $l$ | $m$ | $n$ | $o$ | $p$ | $q$ | $r$ | $s$ | $t$ | $u$ | $v$ | $w$ | $x$ | $y$ | $z$ | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ | $g$ | $h$ | $i$ | $j$ |

Figure 1.7: Vigenère Cipher Table: row 0 is the key, 1 even, and 2 odd shift.

For example "Coffee" would be "hykpjo" deterring frequency analysis of the letter 'E'.

**Function 2.1: Key & String Length ($\lambda, \|a\|$)**

The rest of the text may denote '$\lambda$' (lambda) as the bit length of the key. E.g., $\lambda = 8$ bits, which in in binary may hold $2^8 = 256$ values ($[0000\ 0000]_2$-$[1111\ 1111]_2$). The $\lambda$ is variable is often called the **security parameter**.

---

**For ease of notation**, $\|a\|$ denotes the length of a character or binary string $a$. E.g., $\|a\| = 5$ for the string $a =$ "hello" (the use of which will always be explicit). [17]

**Definition 2.4: One-Time Pad (OTP)**

OTP, also known as the **Vernam Cipher** is a cryptographic scheme, invented by Gilbert Vernam in 1919. Earlier depictions though date back to 1882 by Frank Miller on telegraphy.

---

**Security Definition**: Confidentiality is guaranteed if the key is used only once.

**Function 2.2: One Time Pad - `Enc(m, k)` & `Dec(k, c)`**

Let function $k \leftarrow \{0,1\}^\lambda$ generate keys $k$:

- $\{0,1\}$ denotes the set of possible inputs.
- $k$ is a random bit string of length $\lambda$ bits consisting of 0's and 1's. E.g., $\lambda = 8$ then $k = [1010\ 1101]_2$ is a possible output.
- $\{0,1\}^\lambda$ should be a uniform distribution, i.e., each bit is equally likely to be 0 or 1.

Let $m$ be and $c$ be an encrypted message, both length $\lambda$ bits. Then:

- $c \leftarrow \text{Enc}(m, k) := m \oplus k$. (Encryption)
- $m \leftarrow \text{Dec}(k, c) := c \oplus k$. (Decryption)

Where $\oplus$ denotes the XOR operation ($1 \oplus 1 = 0; 0 \oplus 1 = 1; 1 \oplus 0 = 1; 0 \oplus 0 = 0$).

$$c \leftarrow \text{Enc}(m,k) := \begin{cases} 0011\ 0100\ 1101\ 1000\ 1111 & (m) \\ \oplus\ 1110\ 1010\ 0110\ 1000\ 1101 & (k) \\ \hline 1101\ 1110\ 1011\ 0000\ 0010 & (c) \end{cases} \quad \text{(Encryption)}$$

$$m \leftarrow \text{Dec}(c,k) := \begin{cases} 1101\ 1110\ 1011\ 0000\ 0010 & (c) \\ \oplus\ 1110\ 1010\ 0110\ 1000\ 1101 & (k) \\ \hline 0011\ 0100\ 1101\ 1000\ 1111 & (m) \end{cases} \quad \text{(Decryption)}$$

The larger the key, the more secure the scheme becomes, as smaller keys have a higher probability of being brute-forced by generating all possible keys.

**Theorem 2.4: Computational Security**

**Computational Security** is a security definition that guarantees that the adversary cannot break the scheme in a reasonable amount of time. In todays standards, exponential time algorithms are considered infeasible, (e.g., $O(2^\lambda)$ time complexity).

| **Efficient algorithm known:** | **No known efficient algorithm:** |
|---|---|
| Computing GCDs | Factoring integers |
| Arithmetic mod $N$ | Computing $\phi(N)$ given $N$ |
| Inverses mod $N$ | Discrete logarithm |
| Exponentiation mod $N$ | Square roots mod composite $N$ |

Figure 1.8: Comparison of problems with known efficient algorithms and those without [17].

**Definition 2.5: Known & Chosen Plaintext Attacks**

**Known Plaintext Attack (KPA)**:
The adversary has access to one or more known unencrypted and encrpyted message pairs.

**Chosen Plaintext Attack (CPA)**:
The adversary encrypts plaintext of their choosing to analyze the corresponding ciphertexts.

The Caesar Cipher and Vigenère Cipher are both vulnerable to plaintext attacks. The One-Time Pad (OTP), becomes vulnerable if the key is small or reused.

**Definition 2.6: Block Ciphers**

A cryptographic scheme that separates and encrypts fixed-length blocks of plaintext into ciphertext. Let $\beta$ (beta) be the block size, and $\lambda$ the key length. Then for a set of $B$ blocks and message $M$: $\sum_{b \in B} \beta = \|M\|$ (the sum of all blocks equals the length of the message).
We define $\text{Enc}_\lambda(M, \beta)$, $\text{Dec}_\lambda(C, \beta)$, $C$ as the set of ciphertext blocks, s.t.:

$$\text{Enc}_\lambda(M, \beta) \to C : b \in B \mapsto c \in C$$
$$\text{Dec}_\lambda(C, \beta) \to M : c \in C \mapsto b \in B$$

Where $\lambda := \{(b, c), \dots\}$, a dictionary of unique message block to ciphertext pairs (bijection).

The below figure demonstrates use of a block cipher with a block size of 2, using the Electronic Codebook Mode (ECB).

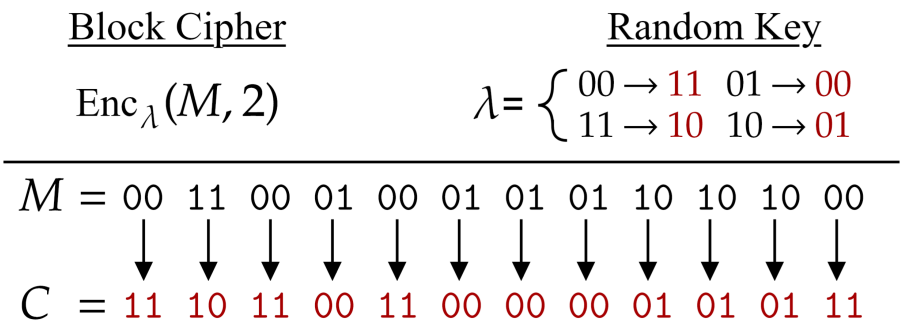



Figure 1.9: Electronic Codebook Mode (ECB) with a block size of 2 [12].

Though in its simplicity falls to the same weakness as the Caesar Cipher, as identical plaintext blocks will encrypt to the same ciphertext block.

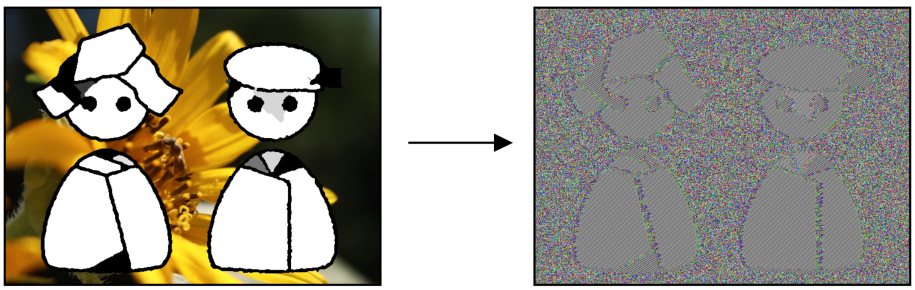


Figure 1.10: An unencrypted image (left) and the same image encrypted with ECB (right).

Here the image on the right is still partially recognizable, as when ECB encountered white space, it encrypted it to the same block. Shockingly the popular video conferencing software Zoom used ECB during the 2020 COVID-19 pandemic, of which now has been patched.

**Definition 2.7: Cipher Block Chaining (CBC)**

CBC encrypts blocks of plaintext into ciphertext. CBC uses a **Initialization Vector (IV)** to start the chain of XORing. The IV is XORed with the first plaintext block. Each XOR is indexed into the key value pair dictionary $\lambda$. The result is the cypher text block. Then the ciphertext block XORs with the next plaintext block and so on. [11]

**Security Definition**: Confidentiality. Integrity and Authenticity are not guaranteed.

The below figure demonstrates use of a block cipher with the Cipher Block Chaining Mode (CBC).

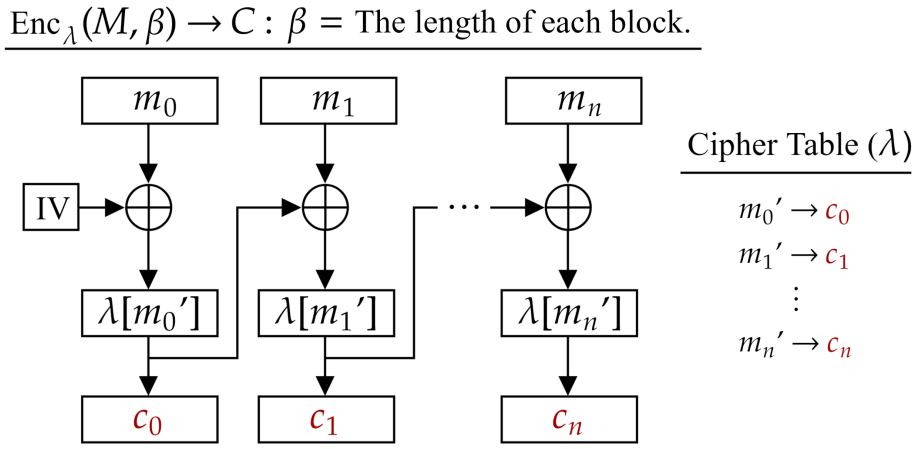



Figure 1.11: A block cipher utilizing the Cipher Block Chaining Mode (CBC) method. [12].

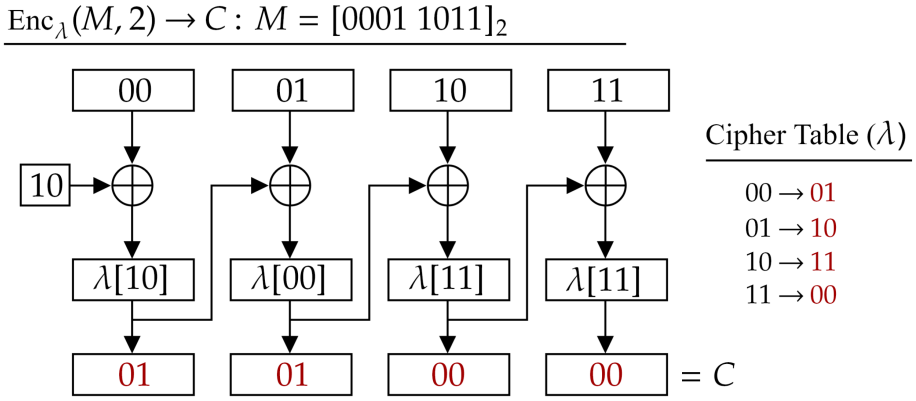



Figure 1.12: CBC encryption example with input $[0001\ 1011]_2$ and outputs $[0101\ 0000]_2$.

Plenty of block cyphers elaborate on these concepts. The different flavors are called a **mode of operation**.

**Definition 2.8: Message Authentication Code (MAC)**

A MAC combines a message with a secret key before hashing. Let $M$ be the message, $\lambda$ the key, and the function $T \leftarrow \text{Enc}_\lambda(M)$. Where $T$ is a resulting tag, sometimes called a **digest** or **hash**. Then $(M, T)$ is sent over the wire. The receiver also has $\lambda$ and runs $\text{Enc}_\lambda(M)$ to verify $T$.

---

**Security Definition**: Integrity and Authenticity. Not confidentiality.

The following figure demonstrates that a MAC protects integrity, as if the message were altered, the receiver would not get the same tag with their key. Though an adversary may still intercept and read the message.

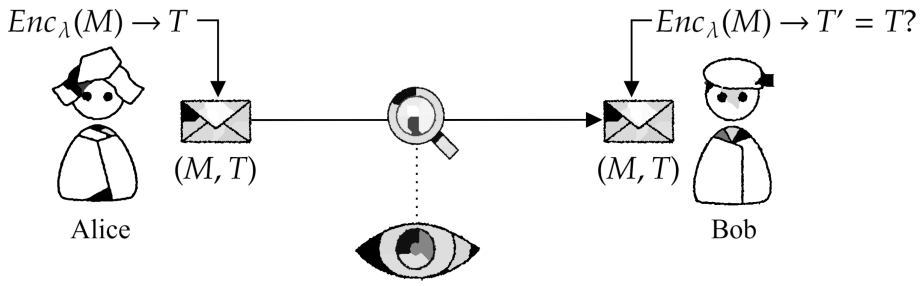



Figure 1.13: A MAC protecting integrity.

**Theorem 2.5: Replay Attack**

A replay attack is when an adversary intercepts a message and resends it to the receiver. The receiver may not know the message was sent twice.

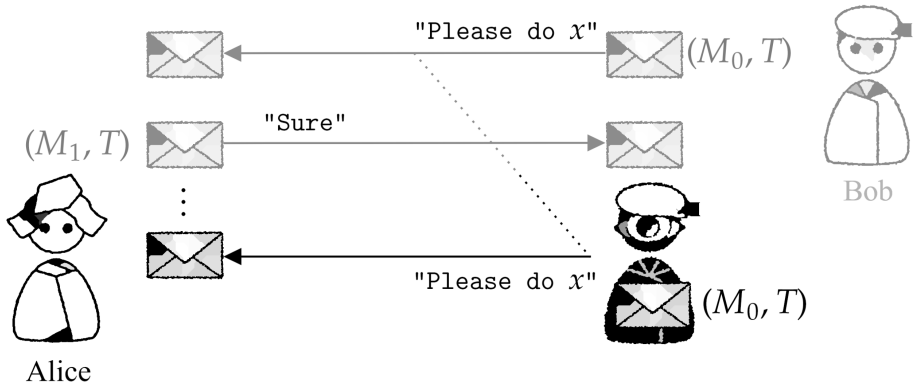



Figure 1.14: A replay attack, where the adversary intercepts and resends a message.

**Theorem 2.6: Replay Attack Prevention**

To prevent replay attacks, a timestamp or nonce (number used once) is added to the message. The receiver checks the timestamp or nonce to ensure the message is fresh.

**Definition 2.9: Hashed-based Message Authentication Code (HMAC)**

HMACs are a type of MAC which are standardized and deemed secure. They take a predefined hash function (e.g., SHA-256) and apply it to a MAC. I.e., an HMAC is a specific recipe for a MAC. [18]

**Definition 2.10: Galios Counter Mode (GCM)**

GCM uses a MAC called GMAC, which is a variant of the HMAC. It encrypts data with a desired Encryption Algorithm and then GMACs (MACs) the data into cipher text. The final hash is the tag. This process goes as follows:

- **Encryption**:
  1. Each block of plaintext is XORed with the encryption key.
  2. To ensure randomness, a counter is added to each encryption before XORing.
  3. To ensure randomness of the counter, an IV is added to it.
- **MAC**:
  1. After encryption, the data is XORed with a previous hash then GMACed.
  2. The GMAC is then used to XOR the next block's cipher before hashing again.
  3. To start the chain of XORing, a 128-bits of zeros is encrypted and GMACed.
  4. After the final block, the length of the message is XORed with the prior hash, then GMACed.
  5. Finally, an encryption of 32-bits of zeros is XORed with the final hash, producing the tag.

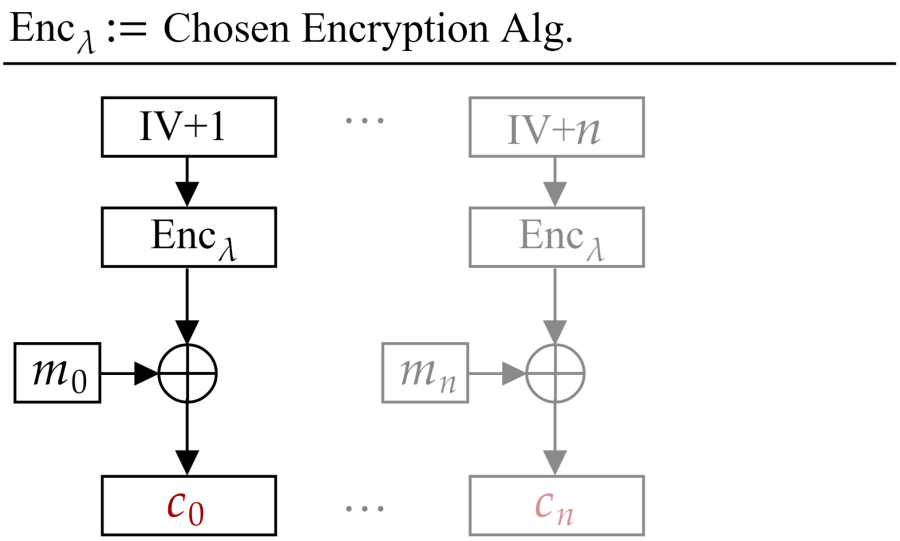
GCM also supports **Authentication of Additional Data (AAD)**, which is data that is not encrypted but is still hashed. So in addition to authenticating and encrypting a message, GCM can also authenticate additional unencrypted data. If GCM is only used for authentication, it is called **GMAC**. [20]

---

**Security Definition**: Confidentiality, Integrity, and Authenticity.

*GCM Diagram and Elaborations on the next page.*

The following figures are first broken up and then combined for clarity.




1em

Figure 1.15: GCM IV and Counter XORing with Plaintext to create Ciphertext.

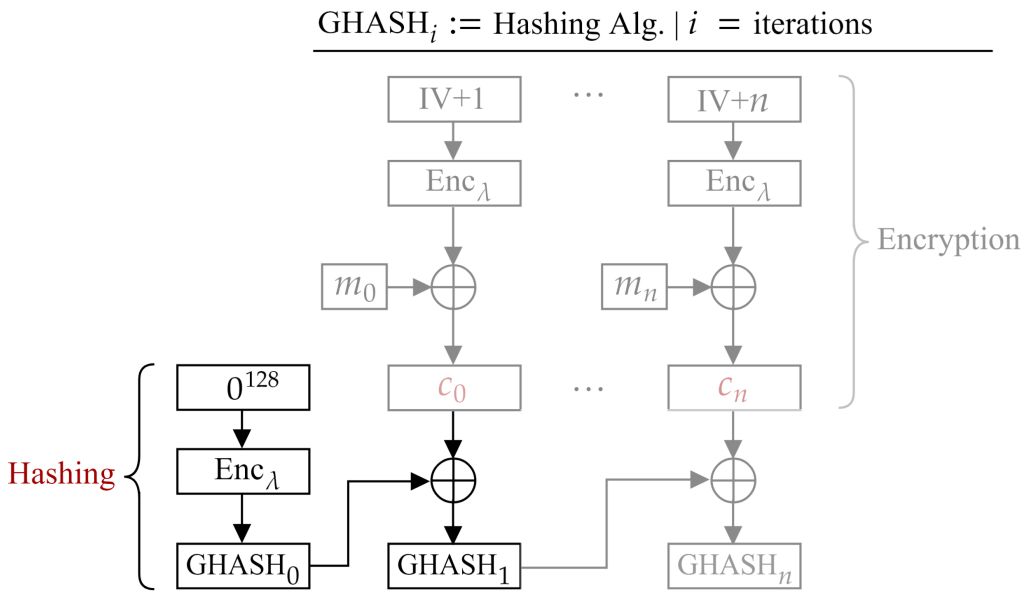



Figure 1.16: 128-bits of zeros is encrypted and GMACed, starting the chain of XOR hashing.

*Continued on the next page.*

To finish the chain of XOR hashing:

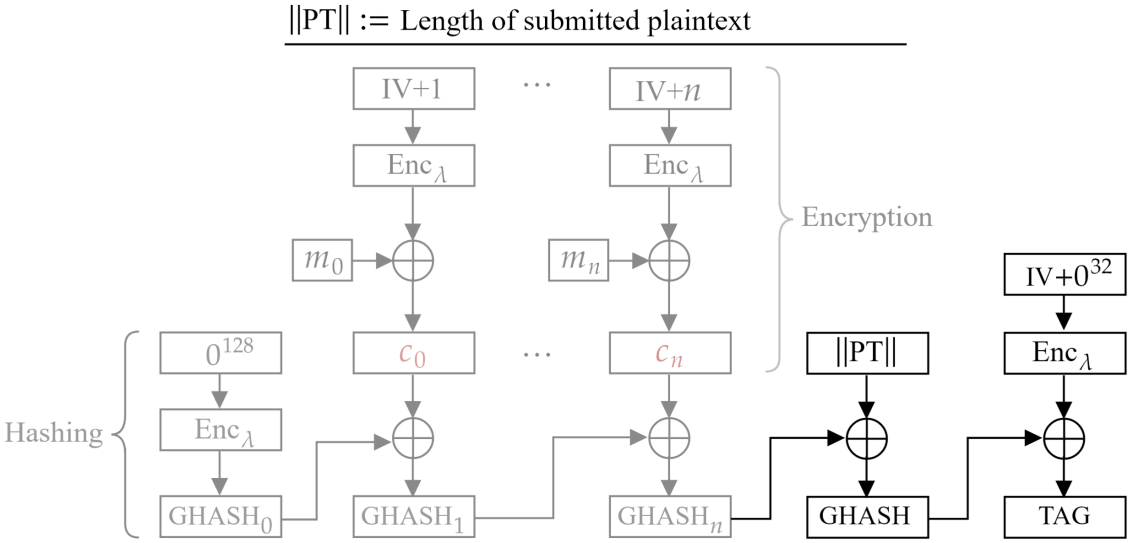



Figure 1.17: The length of the message is XORed with the prior hash, then GMACed with 32-bits of encrypted zeros concatenated with the IV.

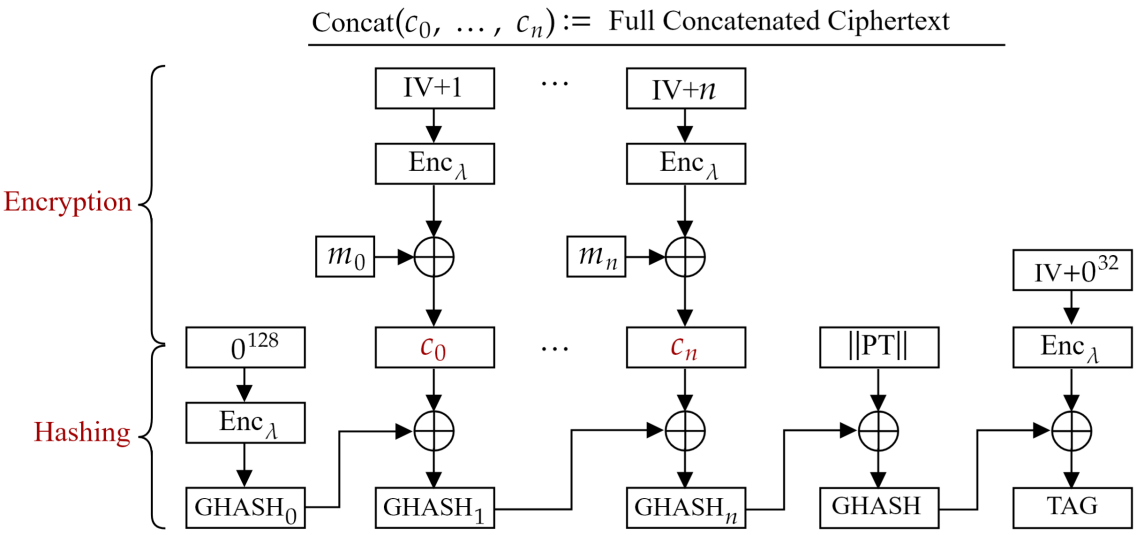



Figure 1.18: A semi complete GCM diagram (Encryption and Authentication), with AAD left out.

*Continued on the next page.*

A full GCM diagram with AAD:

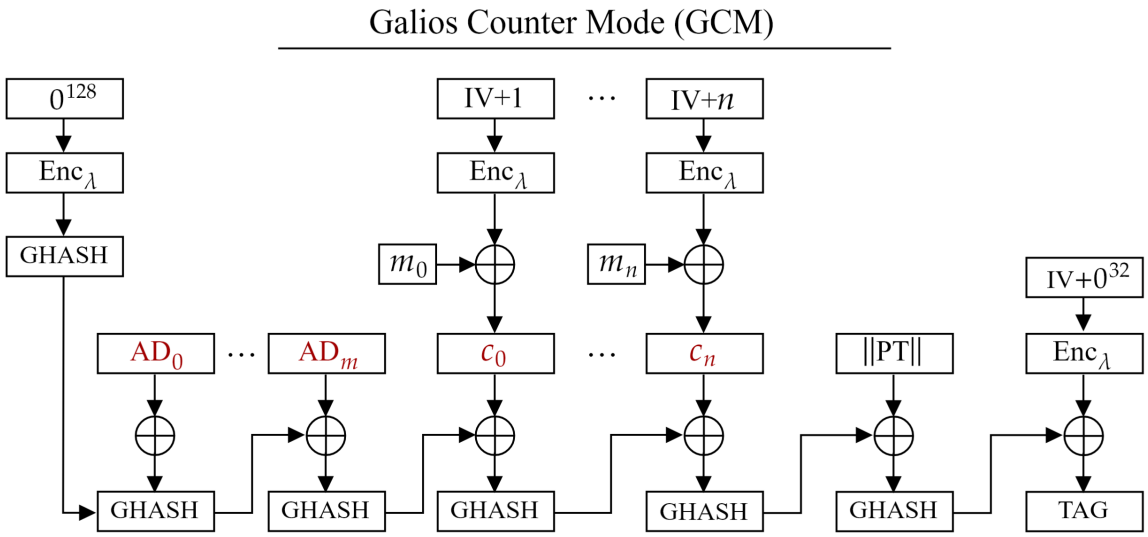



Figure 1.19: A full GCM diagram: Encryption, Authentication, and unencrypted Additional Authenticated Data (AAD).

Now to discuss a possible shared key (symmetric key) algorithm usable Enc$_\lambda$ of GCM.

**Definition 2.11: Advanced Encryption Standard (AES)**

During a worldwide competition in 2001, the National Institute of Standards and Technology (NIST) selected the Rijndael algorithm as the Advanced Encryption Standard (AES). AES is a symmetric key block cipher that encrypts plaintext (PT) in 128-bit blocks. AES works in rounds permuting the PT with partial keys generated from the initial key.

- **Key Expansion**: An initial key is expanded into a key schedule. These keys will be assigned to each round. The rounds needed depend on the initial key size:
  1. 128-bit key: 10 rounds, 11 keys.
  2. 192-bit key: 12 rounds, 13 keys.
  3. 256-bit key: 14 rounds, 15 keys.
- **Input Transformation**: The PT is transformed into a $4 \times 4$ matrix, called the **State**. The state undergoes four main operations per round:
  1. **SubBytes**: Each byte is substituted with a value from the S-Box.
  2. **ShiftRows**: Each row is shifted left by an offset.
  3. **MixColumns**: Each column is mixed with a fixed matrix.
  4. **AddRoundKey**: Each byte in the State is XORed with a sub-key. [3] [14] [1]

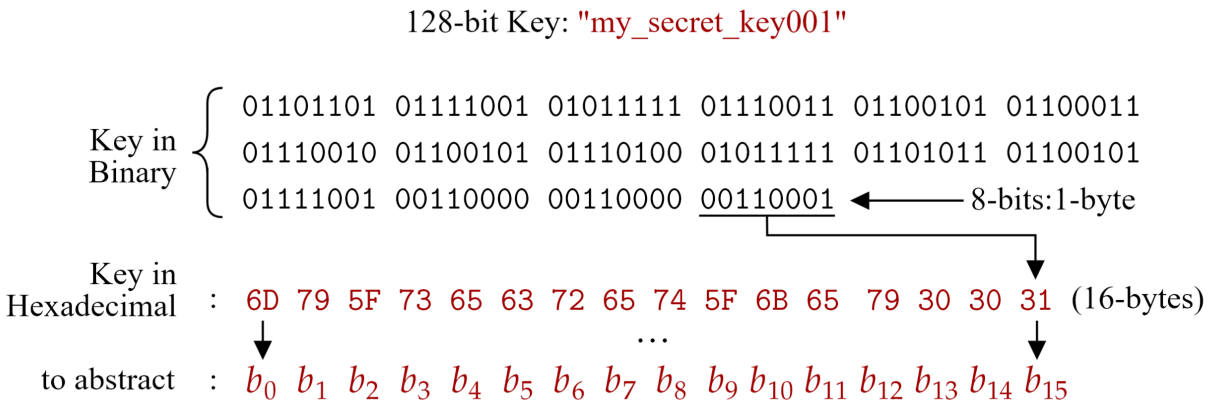
**Definition 2.12: Key Expansion**

Key expansion, an AES process which takes a single key and expands it into multiple keys. The initial key is broken up into 16-byte $4 \times 4$ matrices. Each column a **Word** (32-bits). The process follows four main steps:

**RotWord→SubWord→Rcon→XOR**.
(Rotate Word, Substitute Word, Round Constant, XOR)

This generates a **Sub-Key** for one round. Each round generates for the next round.

**AES Key Expansion:** Given the an initial 128-bit key, $\lambda_0$ "my_secret_key001":




Key represented an a $4 \times 4$ matrix

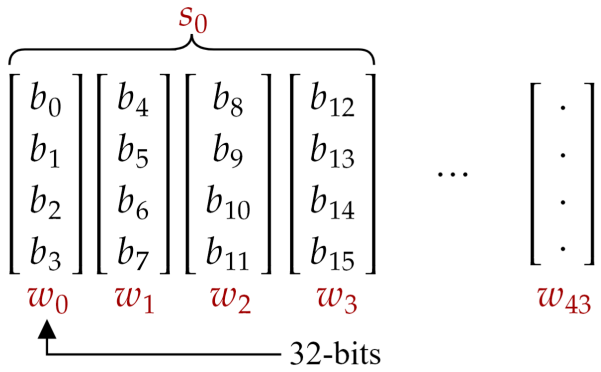
$$\begin{bmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{13} \\ b_3 & b_7 & b_{11} & b_{15} \end{bmatrix}$$

The Key is broken up into Words

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \begin{bmatrix} b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \begin{bmatrix} b_8 \\ b_9 \\ b_{10} \\ b_{11} \end{bmatrix} \begin{bmatrix} b_{12} \\ b_{13} \\ b_{14} \\ b_{15} \end{bmatrix}$$
$$w_0 \quad w_1 \quad w_2 \quad w_3$$

These Words $w_0, w_1, w_2, w_3$ are the initial key. This will then generate the rest of the Words needed. This first group is the first sub-key $s_0$ for the first round.

This is a Subkey ($s_0$), used for a round. The number of rounds depend on the initial Key.



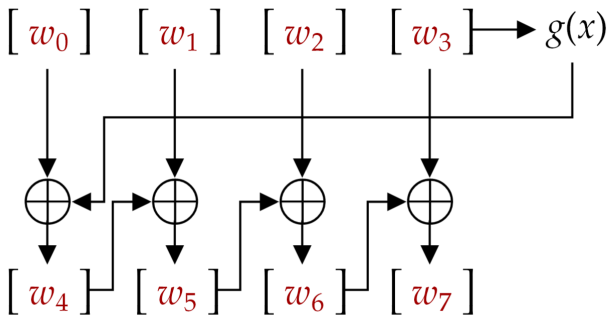

Round Dependence:

- 128-bit key: 10 rounds, 11 keys.
- 192-bit key: 12 rounds, 13 keys.
- 256-bit key: 14 rounds, 15 keys.

Hence, $4 \times 11 = 44$ Words

Rounds depend on $\|\lambda_0\|$: 128-bits $\rightarrow$ 10 rounds, 192-bits $\rightarrow$ 12 rounds, 256-bits $\rightarrow$ 14 rounds.

There need be a total of 44 Words. Expansion is used to find the rest:



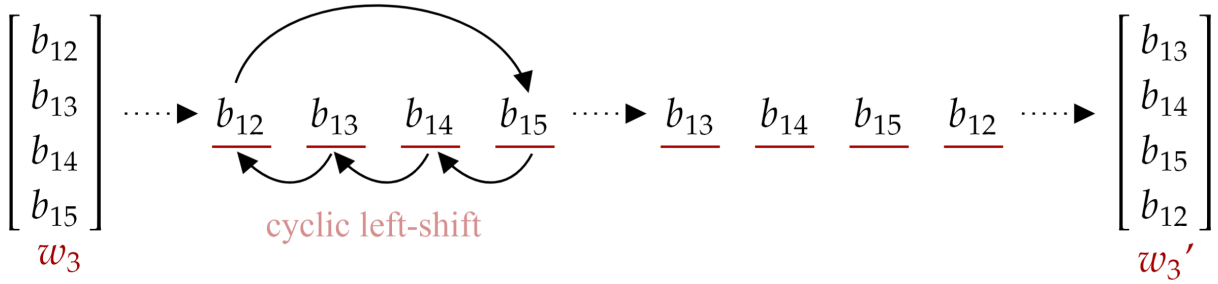

Here $s_0$ generates $s_1$

And so forth $s_i$ generates $s_{i+1}$
Until all Words are satisfied

And $g(x)$ some permuting function

The $g(x) := RotWord \circ SubWord \circ Rcon \circ XOR$, generating the next sub-key $s_1$.

Here, $g(x)$ performs a cyclic left-shift on $w_3$; Often called $RotWord(x)$ (rotate Word).



**Definition 2.13: Nibble**

A **nibble** is a four-bit aggregation, referring to "half a **byte**." A nibble splits 8-bits into two 4-bit values, the **upper nibble** and the **lower nibble**. E.g., given the hex value $0A$, the upper and lower nibble is 0 and $A$ respectively.

Now to quickly introduce some concepts needed for the AES borrowed from group theory.

**Definition 2.14: Field**

A **field** is a set of elements equipped with two operations, addition and multiplication, such that the following properties hold:

- **Closure**: Adding or multiplying any two elements in the set remains in the set.
- **Associativity**: Addition and multiplication are associative.
- **Commutativity**: Addition and multiplication are commutative.
- **Identities**: There exist identity elements for addition (0) and multiplication (1).
- **Inverses**: Every element has an additive inverse (negative) and, except for 0, a multiplicative inverse (reciprocal).
- **Distributivity**: Multiplication distributes over addition.

E.g., The set of real numbers $\mathbb{R}$, with standard addition and multiplication, forms a field.

**Theorem 2.7: Abel Ruffini Theorem**

There is no general for solving polynomials of degree five or higher. Moreover, no formula is possible using these finite amount of: $+, -, \times, \div, \sqrt[n]{x}$.

This will play a key part in the AES algorithm, stopping attackers from reversing the encryption process.

**Definition 2.15: Galois Field $GF(2^8)$**

The Galois Field $GF(2^8)$ is a finite field consisting of $2^8 = 256$ elements, where each element is an 8-bit binary value (a byte). Addition and multiplication in $GF(2^8)$ are defined modulo an irreducible polynomial of degree 8 over $GF(2)$ (the binary field).
In AES, the irreducible polynomial used is:

$$p(x) = x^8 + x^4 + x^3 + x + 1.$$

**Tip:** To learn more consider this video on Galois Theory: "Why is there no quintic formula?"

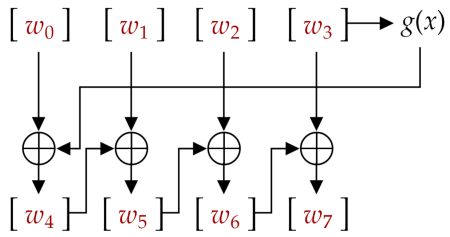
**Definition 2.16: AES S-Box**

The AES S-Box is a substitution box used in the AES algorithm. It is a $16 \times 16$ matrix of 8-bit values. The first column and row represent the upper and lower nibble of the input byte. E.g., given $C7$, column $c0$ and row 07 intersect $c6$.

| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0a | 0b | 0c | 0d | 0e | 0f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| 10 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| 20 | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| 30 | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| 40 | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| 50 | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| 60 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| 70 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| 80 | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| 90 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| a0 | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| b0 | e7 | c8 | 37 | 6d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 | ba |
| c0 | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| d0 | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | 89 | 86 | c1 | 1d | 9e |
| e0 | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| f0 | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

For reference, previous figures are shown again.

There need be a total of 44 Words. Expansion is used to find the rest:

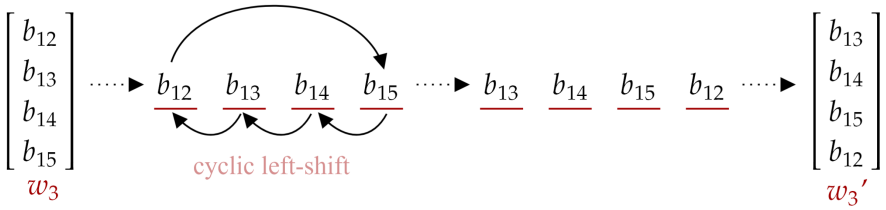



Here $s_0$ generates $s_1$

And so forth $s_i$ generates $s_{i+1}$
Until all Words are satisfied

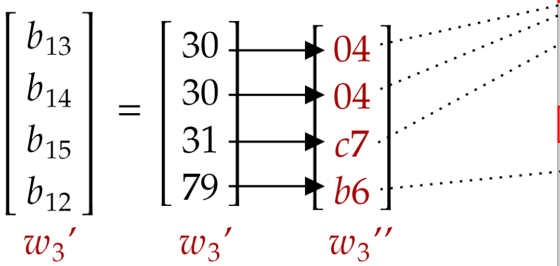
And $g(x)$ some permuting function

Here, $g(x)$ performs a cyclic left-shift on $w_3$; Often called $RotWord(x)$ (rotate Word).




Next $SubWord(x)$ takes the Hex digit places of $b_i$ to index the **AES S-Box**

Key Hex : 6D 79 5F 73 65 63 72 65 74 5F 6B 65 79 30 30 31




| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0a | 0b | 0c | 0d | 0e | 0f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| 10 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| 20 | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| 30 | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| 40 | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| 50 | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| 60 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| 70 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| 80 | cd | 0e | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| 90 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| a0 | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| b0 | e7 | c8 | 37 | 6d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 | ba |
| c0 | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| d0 | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | 89 | 86 | c1 | 1d | 9e |
| e0 | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| f0 | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

**Definition 2.17: Round Constants**

Round constants take the output of $SubWord(x)$ and XOR the first byte with a constant value. The constant value, denoted $Rcon[j]$, is derived from powers of 2 in the Galois Field $GF(2^8)$, where the irreducible polynomial is $x^8 + x^4 + x^3 + x + 1$.

Each round constant ensures the uniqueness of round keys during the AES key expansion process, introducing nonlinearity and breaking symmetry between rounds.

| Round (i) | *Rcon* Value | Power of 2 in GF($2^8$) |
|---|---|---|
| 1 | 0x01 00 00 00 | $2^0 = 1$ |
| 2 | 0x02 00 00 00 | $2^1 = 2$ |
| 3 | 0x04 00 00 00 | $2^2 = 4$ |
| 4 | 0x08 00 00 00 | $2^3 = 8$ |
| 5 | 0x10 00 00 00 | $2^4 = 16$ |
| 6 | 0x20 00 00 00 | $2^5 = 32$ |
| 7 | 0x40 00 00 00 | $2^6 = 64$ |
| 8 | 0x80 00 00 00 | $2^7 = 128$ |
| 9 | 0x1B 00 00 00 | $2^8$ mod poly |
| 10 | 0x36 00 00 00 | $2^9$ mod poly |

Table 1.1: Round Constants, where poly:= $x^8 + x^4 + x^3 + x + 1$.

Finally $w_3''$ is XORed with it's respective round constant $Rcon[j]$ from $GF(2^8)$.

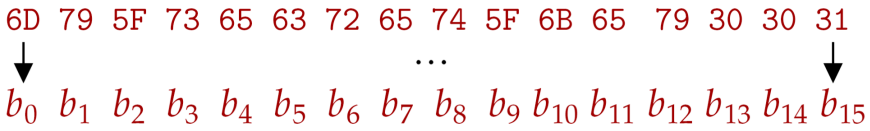
$$
\begin{aligned}
w_3'' &= 0000\ 0100\ 0000\ 0100\ 1100\ 0111\ 1011\ 0110 \\
Rcon[1] &= 0000\ 0001\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000 \\
\hline
w_3'' \oplus Rcon[1] &= 0000\ 0101\ 0000\ 0100\ 1100\ 0111\ 1011\ 0110
\end{aligned}
\quad \rightarrow \texttt{05 04 C7 B6}
$$

$$
= \begin{bmatrix} 05 \\ 04 \\ C7 \\ B6 \end{bmatrix}
$$

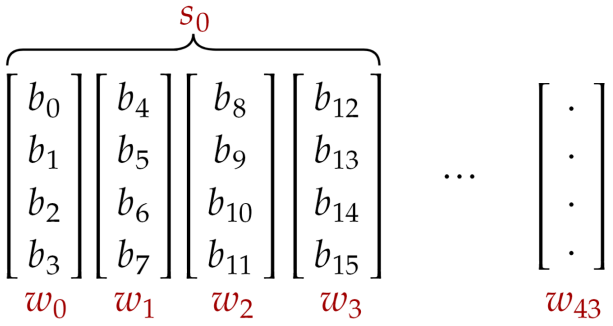
$g(w_3)$

All together:

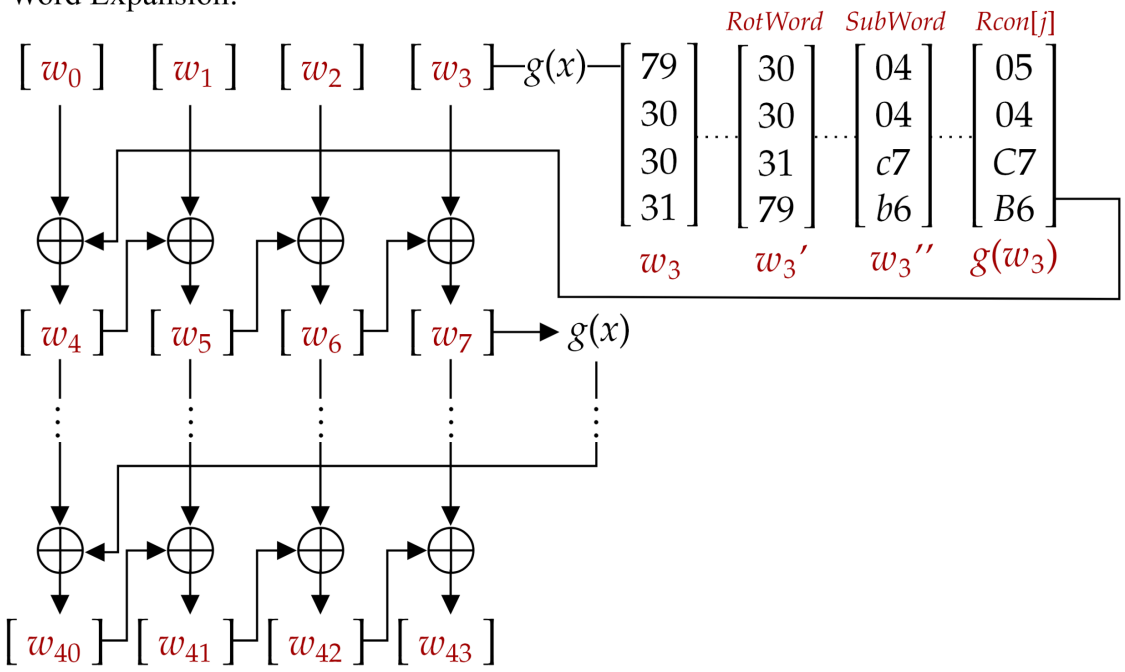
AES Key Expansion of a 128-bit Key: "my_secret_key001". First, Hex conversion:




Word Aggregation into Subkeys:



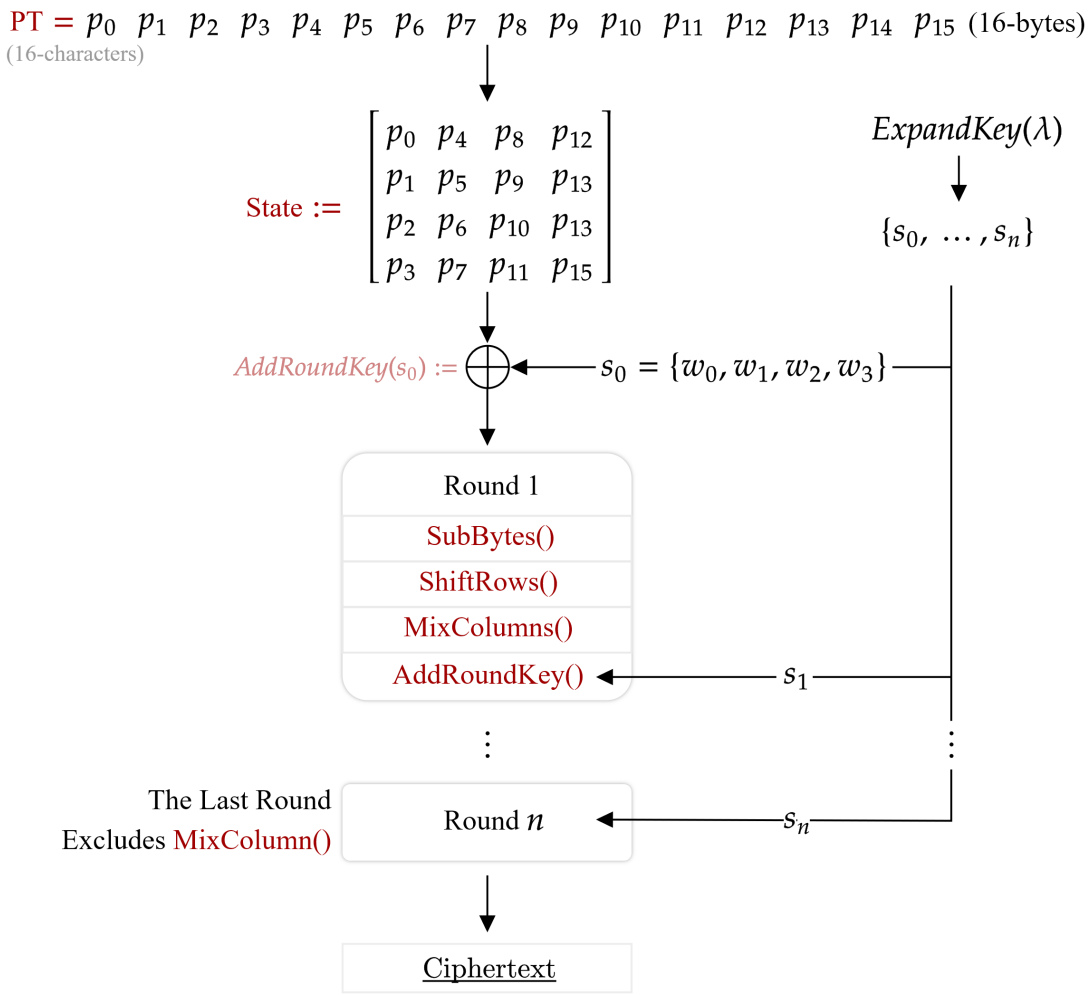

Word Expansion:




This is AES Key Expansion.

Now to discuss the AES encryption process.

The Four main steps in AES Encryption of a 128-bit Plaintext (PT) with a 128-bit key ($\lambda$).




First the PT is transformed into a $4 \times 4$ matrix, called the **State**. The initial transformation is the **AddRoundKey** operation, which XORs each column of the state with sub-key $s_0 = \{w_0, w_1, w_2, w_3\}$. I.e., $[p_0, p_1, p_2, p_3] \oplus [w_0]$ and so on. Recall $w_0 = [b_0, b_1, b_2, b_3]$ for some byte $b_i$ of the initial key.

To elaborate on the AddRoundKey operation:

The AddRoundKey() XORs each column $c_i$ with a sub-key Word $w_i$

PT = "plain_text_mssg0" Hex = `70 6C 61 69 6E 5F 74 65 78 74 5F 6D 73 73 67 30`

$$\text{State} := \begin{bmatrix} c_0 & c_1 & c_2 & c_3 \\ 70 & 6E & 78 & 73 \\ 6C & 5F & 74 & 73 \\ 61 & 74 & 5F & 67 \\ 69 & 65 & 6D & 30 \end{bmatrix} \oplus \begin{bmatrix} w_0 & w_1 & w_2 & w_3 \\ b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{13} \\ b_3 & b_7 & b_{11} & b_{15} \end{bmatrix} = \begin{bmatrix} p_0' & p_4' & p_8' & p_{12}' \\ p_1' & p_5' & p_9' & p_{13}' \\ p_2' & p_6' & p_{10}' & p_{13}' \\ p_3' & p_7' & p_{11}' & p_{15}' \end{bmatrix}$$

$s_0$

$$\begin{matrix} 6E \oplus b_4 \\ \vdots \\ 65 \oplus b_7 \end{matrix}$$

AddRoundKey takes a sub-key $s_i$ and XORs each Word $w_i$ with the corresponding column $c_i$ of the State.

The SubBytes() performs an **AES S-Box** substitution with every byte of the State.

$$\begin{bmatrix} p_0' & p_4' & p_8' & p_{12}' \\ p_1' & p_5' & p_9' & p_{13}' \\ p_2' & p_6' & p_{10}' & p_{13}' \\ p_3' & p_7' & p_{11}' & p_{15}' \end{bmatrix} \rightarrow \boxed{\text{AES S-Box}} \rightarrow \begin{bmatrix} p_0'' & p_4'' & p_8'' & p_{12}'' \\ p_1'' & p_5'' & p_9'' & p_{13}'' \\ p_2'' & p_6'' & p_{10}'' & p_{13}'' \\ p_3'' & p_7'' & p_{11}'' & p_{15}'' \end{bmatrix}$$

Just like in Key Expansion's (2.12) use of the S-Box on the last word of each sub-key. The PT undergoes a full substitution with the S-Box.

The ShiftRows() a fixed number of left-shifts depending on the row $r_i$ of the State.

$S(r_i, x) :=$ shifts row $r_i$ to the left by $x$ places (for ease of notation)

$$\begin{bmatrix} p_0'' & p_4'' & p_8'' & p_{12}'' \\ p_1'' & p_5'' & p_9'' & p_{13}'' \\ p_2'' & p_6'' & p_{10}'' & p_{13}'' \\ p_3'' & p_7'' & p_{11}'' & p_{15}'' \end{bmatrix} \begin{matrix} \longrightarrow S(r_0, 0) \longrightarrow \\ \longrightarrow S(r_1, 1) \longrightarrow \\ \longrightarrow S(r_2, 2) \longrightarrow \\ \longrightarrow S(r_3, 3) \longrightarrow \end{matrix} \begin{bmatrix} p_0'' & p_4'' & p_8'' & p_{12}'' \\ p_5'' & p_9'' & p_{13}'' & p_1'' \\ p_{10}'' & p_{13}'' & p_2'' & p_6'' \\ p_{15}'' & p_3'' & p_7'' & p_{11}'' \end{bmatrix}$$

Below, the notation {01} refers to a hex value.

In MixColumns(), Columns $c_i$ are considered four-term polynomials of $GF(2^8)$ and multiplied by polynomial $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$

$$\vdots$$

$$= \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \quad \text{(polynomial matrix representation)}$$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} p_0'' \\ p_5'' \\ p_{10}'' \\ p_{15}'' \end{bmatrix} = \begin{bmatrix} p_0''' \\ p_5''' \\ p_{10}''' \\ p_{15}''' \end{bmatrix} \cdots\cdots$$

$$\vdots$$

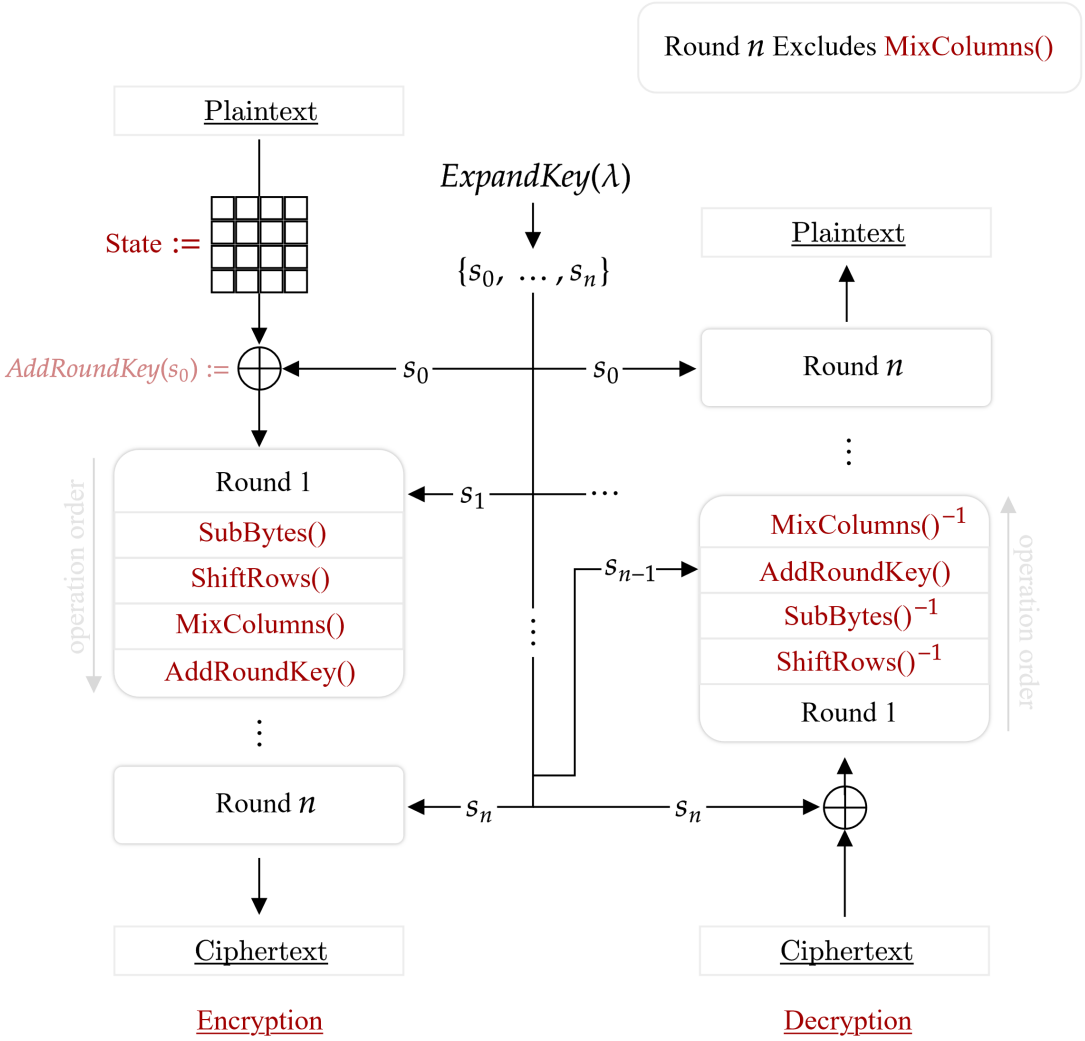
$$p_0''' = (\{02\} \cdot p_0'') \oplus (\{03\} \cdot p_5'') \oplus p_{10}'' \oplus p_{15}''$$

$G(2^8)$, $a + b := a \oplus b$ $(XOR)$

---

$$\begin{bmatrix} p_0'' & p_4'' & p_8'' & p_{12}'' \\ p_5'' & p_9'' & p_{13}'' & p_1'' \\ p_{10}'' & p_{13}'' & p_2'' & p_6'' \\ p_{15}'' & p_3'' & p_7'' & p_{11}'' \end{bmatrix} \longrightarrow \text{MixColumns()} \longrightarrow \begin{bmatrix} p_0''' & p_4''' & p_8''' & p_{12}''' \\ p_5''' & p_9''' & p_{13}''' & p_1''' \\ p_{10}''' & p_{13}''' & p_2''' & p_6''' \\ p_{15}''' & p_3''' & p_7''' & p_{11}''' \end{bmatrix}$$

This is the AES Encryption & Decryption.




For Decryption, the last scheduled key for Encryption is used as the initial transformation. Note the inverse operations of Decryption follow a different order than Encryption. This covers the basics of AES.

# Bibliography


[1] Aes (advanced encryption standard) structure, 2018–2023. Copyright © 2018–2023 BrainKart.com; Developed by Therithal info, Chennai. Accessed: 2024-06-15.

[2] Daniel Adetunji. Symmetric and asymmetric key encryption – explained in plain english, April 2023. Accessed: 2024-12-14.

[3] Satish CJ. Aes iii - advanced encryption standard - introduction, key expansion in aes cyber security cse4003, 2024. Accessed: 2024-06-15.

[4] Cloudflare. Dns over tls vs. dns over https | secure dns. https://www.cloudflare.com/learning/dns/dns-over-tls/, n.d. Accessed: 2024-12-14.

[5] Cloudflare. The dnssec root signing ceremony. https://www.cloudflare.com/learning/dns/dnssec/root-signing-ceremony/, n.d. Accessed: 2024-12-14.

[6] Cloudflare. What is tls (transport layer security)? https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/, n.d. Accessed: 2024-12-14.

[7] Cloudflare. Why is http not secure? | http vs. https. https://www.cloudflare.com/learning/ssl/why-is-http-not-secure/, n.d. Accessed: 2024-12-14.

[8] Codecademy. What is hashing, and how does it work? https://www.codecademy.com/resources/blog/what-is-hashing/, April 2023. Accessed: 2024-12-14.

[9] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. Request for Comments (RFC) 5280, May 2008. Obsoletes RFC 3280, RFC 4325, RFC 4630.

[10] Cybersecurity and Infrastructure Security Agency (CISA). Understanding digital signatures. https://www.cisa.gov/news-events/news/understanding-digital-signatures, February 2021. Accessed: 2024-12-14.

[11] Morris Dworkin. Recommendation for block cipher modes of operation: Methods and techniques. Special Publication 800-38A, National Institute of Standards and Technology (NIST), 2001.

[12] Aleksander Essex. Encrypting with block ciphers, 2024. YouTube video.

[13] Kinsta. Tls vs ssl: What's the difference? which one should you use? https://kinsta.com/knowledgebase/tls-vs-ssl/, 2019. Published December 19, 2019. Updated August 14, 2023. Accessed: 2024-12-14.

[14] National Institute of Standards and Technology (NIST), Morris J. Dworkin, Elaine Barker, James R. Nechvatal, James Foti, Lawrence E. Bassham, E. Roback, and James F. Dray Jr. Advanced encryption standard (aes). Federal Information Processing Standards Publication

197, National Institute of Standards and Technology, November 2001. Accessed: 2024-06-15. Local download available at https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=901427.

[15] M. Nystrom and B. Kaliski. Pkcs #10: Certification request syntax specification version 1.7. Request for Comments (RFC) 2986, November 2000. Obsoletes RFC 2314.

[16] Okta. What is public key infrastructure (pki) and how does it work? https://www.okta.com/identity-101/public-key-infrastructure/, August 2024. Accessed: 2024-12-14.

[17] Mike Rosulek. The joy of cryptography. https://joyofcryptography.com.

[18] Seth. What is the difference between mac and hmac? Crypto Stack Exchange, 2013. Answered on Mar 1, 2013, edited Feb 9, 2016. Available at: https://crypto.stackexchange.com/questions/6523/what-is-the-difference-between-mac-and-hmac [Accessed: YYYY-MM-DD].

[19] Forrest Timour. An abridged history of cryptography, 2019. Available online.

[20] Vidder, Inc. Galois/counter mode (gcm) and gmac, 2016. Accessed: 2024-06-15.

[21] Concise Works. Sect modules. https://github.com/Concise-Works/sect-modules, n.d. Accessed November 2024.

[22] Tal Yitzhak. Understanding digital certificates: Self-signed vs. ca-signed certificates. https://medium.com/@talyitzhak/understanding-digital-certificates-and-self-signed-certificates-b1cdca759bbc, July 2024. Accessed: 2024-12-14.