# BGK501E

Ömer Üstün

November 4, 2023

## Introduction to Network Security

Possible threats to network security:

- DDoS attacks
- Unauthorized access to servers, computers, and data
- Malware injection
- Unauthorized data modification and deletion
- Unauthorized usage of resources
- Damage to customers
- Usage of social engineering
- Advanced persistent threats (APTs)

The attack may come from other trusted networks or it may come from inside.

**C** - Confidentiality

**I** - Integrity

**A** - Availability

### Threats to Confidentiality:

- Unauthorized access to network
- Malicious software injection
- Unauthorized access to data stored in the network
- Unauthorized access to the data in transit

### Threats to Integrity:

- Malicious software injection
- Unauthorized modification/deletion of data
- Unauthorized changes to the configuration of network devices and servers

### Threats to Availability:

- DDoS attacks
- Malicious software injections

- Unauthorized modification/deletion of data
- Data availability attacks (i.e., ransomware)
- Unauthorized changes to the configuration of network devices and servers

Consider the capability of the adversaries and protection requirements because deploying everything you got will be very expensive, add unnecessary cost to each protection mechanism, bring difficulties to perform daily business operations.

You have to strike a balance between protection and availability.

## Different classifications of adversaries:

- Active/passive adversaries
- Inside/outside adversaries
- Purpose of possible attacks

## Protection requirements for assets:

- Confidentiality
- Integrity
- Replay protection
- Non-repudiation
- Prevention of malicious usage
- Prevention of unauthorized access

If there's no sensitive data to protect then there's no need to introduce overhead by encryption.

The general approach to network security is to take all the input and output doors to the network under control.

## Defense mechanisms in network security:

- Physical
  - Perimeter security
- Software/hardware
  - Firewall
  - WAF
  - SIEM
  - IDS/IPS
  - DDoS prevention
  - IDM (identity management)

  - NAC (network access control)
  - EDR (endpoint detection and response)
  - SOAR (security orchestration automation and response)
  - Vulnerability management
  - Pentest
  - Policies, procedures
  - Compliance checks
  - Anomaly detection
- Cryptography and secure communication

## Questions and Answers:

**Q:** Since we have these many tools that protect our network, is it still necessary to use cryptography?

**A:** Yes, it is needed. Because when the data leaves the network it needs to be protected. It also needs to be protected when it is in transit in the network and of course it needs to be protected when at rest.

Cryptography and secure communication provide confidentiality, integrity, non-repudiation, authentication.

**Q:** If everyone is the receiver, is it necessary to establish a secure channel? (Clue: Consider GPS)

We need to define the risks, the adversaries, and the adversaries' capabilities. Define what we need. Define what the adversaries can do.

## Potential of adversaries:

- Passive attacks:
  - Unauthorized access to information (confidentiality)
  - Traffic analysis
- Active attacks:
  - Impersonation
  - Replay attack (A malicious actor intercepting and retransmitting data to produce an unauthorized effect, often deceiving a system into granting access or performing an unintended action.)
  - Message modification
  - Denial of service (by disrupting the communication)
  - Repudiation

### Security solutions on network layers:

TCP/IP divides the networking function into 4 layers:

- Application layer: PGP
- Transport layer: TLS
- Network layer: IPSec
- Datalink layer

# Introduction to Cryptology

Cryptologyy: Practice and study of techniques based on mathematical and computational problems for secure communication. It alone is not enough for security but it is an essential tool.

1. **Discrete Logarithm Problem** (DLP):
   - **What it is:** Given a base $g$, a modulus $p$, and a result $h$, the discrete logarithm problem is to find an exponent $x$ such that $g^x \equiv h \pmod{p}$. In simpler terms, if you know $g$ raised to some power $x$ gives you $h$, the problem is to find that power $x$.
   - **Why it's important:** DLP is the foundation for many cryptographic protocols, including the Diffie-Hellman key exchange and the Digital Signature Algorithm (DSA). The difficulty of solving the DLP provides the security behind these protocols.
2. **Integer Factorization Problem**:
   - **What it is:** Given a composite number $n$, the integer factorization problem involves expressing $n$ as a product of its prime factors.
   - **Why it's important:** The security of the widely used RSA encryption algorithm relies on the difficulty of factorizing large integers. If you can factorize the RSA modulus, you can derive the private key from the public key and break the encryption.
3. **Solution of Complex Equation Systems**:
   - **What it is:** This involves finding solutions to systems of equations that may have multiple variables and can be nonlinear. These equations can be polynomial, exponential, or of other forms.
   - **Why it's important:** Solving complex equation systems has applications in various fields, including optimization, computer algebra, and some cryptographic attacks. The difficulty of solving these systems can also be used as a basis for cryptographic security in certain scenarios.

In the context of cryptography, these problems are computationally hard to solve, especially as the input size grows. This "hardness" provides the security foundation for various cryptographic schemes.

### Use cases:

**Confidentiality:** Only authorized parties can access data.

**Non-repudiation:** The sender cannot dispute the origin of the data.

**Integrity:** Data cannot be modified by unauthorized parties.

**Authentication:** Identification of the entities are proved.

**Secret sharing:** A secret is shared between $n$ parties and any $k$ out of $n$ parties can construct the secret.

**Secure multi-party computation:** Parties want to compute a function using their private data but they don't want to leak their data to each other.

**Privacy:** Parties want to hide their identities and their data while utilizng comm. and computation.

## Cryptology

Cryptology is divided into two main subfields:

**Cryptography:** Design of cryptographic solutions.

**Cryptoanalysis:** Security anaylsis of cryptographic solutions.

## Basic Encryption Model

**Kerckhoffs' Principle (1883):**
*"Security should not rely on the secrecy of the algorithm; everything may be known but the key."*

$$D(k_d, E(k_e, p)) = p$$

$D$**:** Decryption

$E$**:** Encryption

$k_d$**:** Decryption key

$k_e$**:** Encryption key

$p$**:** Plaintext

$c$**:** Ciphertext

## Adversary model

**Questions for modelling:**

- Can the adversary learn the encrypted data?

  *Sniffs the network*

- Can the adversary modify the encrypted data?

  *Sniffs the network, modifies the data and sends the data to the receiver*

- Can the adversary sit between the communicating parties?

  *Changes the proxy configurations in the computer of the victim*

- What is the computation power of the adversary?

  *May have thousands of computation cores. Lets assume one core performs 232 operations in 1 minute. One core can compute* $232 \times 60 \times 24 \times 365 \times 1000 \cong 261$ *operations in 1000 years. The adversary can perform* $261 \times 1000000 \cong 281$ *operations if he/she has 1 million computation cores. That is this adversary can learn the 80-bit secret key of an encryption algorithm by brute-force attack in 250 years in average.*

- Can the adversary learn corresponding ciphertext for the plaintext chosen by himself online or offline?

  *Assume that in a cryptographic protocol, party A sends a challenge (random number) to be encrypted to party B to authenticate party B. The adversary can use party B as an encryption oracle, sends a plaintext and party B returns the corresponding ciphertext.*

## Some adversary models regarding data usable by the adversary

**Known plaintext attack (KPA):** The adversary knows plaintext-ciphertext pairs.

**Ciphertext-only attack (COA):** The adversary only knows ciphertexts.

**Chosen plaintext attack (CPA):** The adversary can learn encryption result of any plaintext chosen by the adversary.

**Chosen ciphertext attack (CCA):** The adversary can learn decryption result of any ciphertext chosen by the adversary.

**Adaptive chosen ciphertext attack (CCA2):** The adversary can learn decryption result of any ciphertext chosen by the adversary during the attack.

**Adaptive chosen plaintext attack (CPA2):** The adversary can learn encryption result of any plaintext chosen by the adversary during the attack.

## Security Definition

**Definition:** An encryption algorithm is said to be **secure** if the difference between:

- The probability of determining partial information about the plaintext for a given ciphertext, and
- The probability of determining partial information about the plaintext without the given ciphertext,

is negligible.

Mathematically, this can be represented as:

$$\Pr[p|c] - \Pr[p] = \varepsilon$$

**In other words:** The adversary should not have any advantage in learning additional information about the plaintext when he knows the ciphertext.

**Illustration:** In an example where the plaintext space consists of alphabetical characters:

- Without the ciphertext, the adversary already knows that the plaintext consists of alphabetical characters.

- Possessing the ciphertext should not increase the adversary's knowledge about the plaintext, assuming the encryption algorithm is secure.

## Historical Encryption Examples

### Shift Ciphers (Caesar Ciphers):

Shift ciphers, one of the earliest encryption techniques, involve shifting each letter in the plaintext by a fixed number of positions in the alphabet.

Given an $n$-letter alphabet, where $p$, $c$, and $k$ are elements of $\mathbb{Z}_n$:

$$E_k(p) = (p + k) \mod n$$
$$D_k(c) = (c - k) \mod n$$

**Security Analysis:**

- Is this cipher secure? It's considered insecure by modern standards due to its simplicity.
- Key Recovery: One can employ an exhaustive key search (brute-force attack). Given that there are only $n$ possible shifts, an attacker can systematically try each shift until the correct decryption is found.

**Substitution Cipher:**

In substitution ciphers, each letter or symbol in the plaintext is replaced by another letter or symbol.

Given $p, c \in \mathbb{Z}_n$; $k$ is a bijection, $f$, over $\mathbb{Z}_n$:

$$E_k(p) = f(p)$$
$$D_k(c) = f^{-1}(c)$$

**Security Analysis:**

- Brute-force? The number of possible keys is $n!$. For a 26-letter alphabet, it is approximately $2^{88}$, making brute-force impractical.
- Frequency Analysis: By computing the distribution of letters in the ciphertext and comparing to known letter distributions in the language, one can deduce likely substitutions.

**Permutation Cipher:**

In permutation ciphers, the letters of the plaintext are rearranged according to a pre-defined system or pattern.

**Security Analysis:**

- The key is essentially the specific permutation used.
- Brute-force? The number of possible keys for 30-letter messages is 30!, approximately $2^{108}$.
- Cryptanalysis: The cipher can potentially be broken using statistics of the language or known plaintext attacks.

**One-Time Pad (Vernam Cipher, 1917):**

The One-Time Pad (OTP) requires a key that is as long as the message and is used only once.

- Shannon's Proof (1949): OTP offers perfect secrecy.
- If Pr $p_c$ − Pr $p = \epsilon$ and this difference is 0, then the encryption has perfect secrecy.
- **Perfect Secrecy:** Every plaintext has equal probability to be encrypted to a given ciphertext, making it information-theoretically secure.

**Computational vs. Information-Theoretic Security:**

- 128-bit AES encryption: Computationally secure.
- One-Time Pad: Information-theoretically secure. Given a ciphertext, the plaintext can be anything from the plaintext space since the key length equals the plaintext length.

**Redundancy Check:**

Redundancy checks, like CRC (Cyclic Redundancy Check), detect and potentially rectify changes in messages.

- Can this be used against modifications of encrypted data by adversaries? No, because adversaries can compute CRC for modified data. Cryptographic solutions, like hash functions, are essential.

**Freshness:**

Ensuring the "freshness" of data can prevent replay attacks.

- Prevention methods: Use timestamps or message sequence numbers.
- Importance: Always define the attack model first and then design a solution for the threat.

# Symmetric Key Cryptography

Symmetric means that the keys used in security-related transformations (encryption, decryption) in both sender and receiver sides are the same.

Both entities must have the key $k$ in a secure way.

**Symmetric Encryption Algorithm Definition:**

Let $\mathcal{K}$, $\mathcal{P}$, $\mathcal{C}$, $E$, and $D$ define the encryption algorithm where:

$$
\begin{aligned}
&E : \mathcal{K} \times \mathcal{P} \to \mathcal{C} \\
&D : \mathcal{K} \times \mathcal{C} \to \mathcal{P} \\
&\forall p \in \mathcal{P}, k \in \mathcal{K} \quad D(k, E(k, p)) = p
\end{aligned}
$$

- $\mathcal{K}$ represents the **key space**. This denotes the set of all potential keys that could be utilized for both encryption and decryption in a symmetric encryption scheme. When we refer to $k \in \mathcal{K}$, it signifies an individual key selected from this key space.
- $\mathcal{P}$ denotes the **plaintext space**, which is the set of all conceivable plaintexts that can be encrypted. An element $p \in \mathcal{P}$ refers to a specific plaintext message from this set.

The notation $\forall p \in \mathcal{P}, k \in \mathcal{K}$ underscores a behavior or property that remains true for every possible combination of plaintext and key. Specifically, in the definition:

$$\forall p \in \mathcal{P}, k \in \mathcal{K} \quad D(k, E(k, p)) = p$$

This states that for every conceivable plaintext $p$ and every possible key $k$, if one encrypts $p$ using $k$ to produce a certain ciphertext, then decrypts that ciphertext using the identical key $k$, the original plaintext $p$ is retrieved. This emphasizes the fundamental correctness of the encryption and decryption operations: they perfectly reverse each other, provided the same key is used for both functions.

## Block Ciphers

Block ciphers are a category of symmetric encryption algorithms that operate on fixed-size blocks of plaintext to produce fixed-size blocks of ciphertext.

- **Block Length ($n$)**: This refers to the size (in bits) of the input data block that the cipher operates on. For instance, a block cipher with a block length of 128 bits operates on 128 bits of plaintext at a time to produce 128 bits of ciphertext.
- **Key Size ($m$)**: The size (in bits) of the key used for encryption and decryption. The size of the key determines the number of possible keys in the key space.
- **Key Space ($\mathcal{K}$)**: The set of all possible keys that can be used for encryption and decryption. Specifically, $\mathcal{K} = \{0,1\}^m$, implying that there are $2^m$ potential keys.
- **Plaintext Space ($\mathcal{P}$) and Ciphertext Space ($\mathcal{C}$)**: Both the plaintext and ciphertext spaces are defined as $\{0,1\}^n$, meaning each plaintext or ciphertext block consists of $n$ bits. Thus, there are $2^n$ possible n-bit blocks.
- **Encryption ($E$) and Decryption ($D$) Functions**:

  $$E : \mathcal{K} \times \mathcal{P} \to \mathcal{C}$$
  $$D : \mathcal{K} \times \mathcal{C} \to \mathcal{P}$$

  The encryption function takes a key and a plaintext block as input and produces a ciphertext block. The decryption function, conversely, takes a key and a ciphertext block and recovers the original plaintext block.
- **Correctness Property**: For every possible key $k$ and every possible plaintext block $p$, decrypting the encryption of $p$ using $k$ should return $p$. Formally, $\forall p \in \mathcal{P}, k \in \mathcal{K} \quad D(k, E(k, p)) = p$.

In practice, when a message is encrypted using a block cipher, it is first divided into $n$-bit blocks (where $n$ is the block size of the encryption algorithm). Each of these blocks is then encrypted individually using the same key $k$. The resulting ciphertext blocks are concatenated to produce the final encrypted message. Decryption operates in a similar manner, decrypting each block separately and then concatenating them to recover the original message.

## Stream Ciphers

Stream ciphers are a type of symmetric encryption algorithm that encrypts plaintext one bit (or sometimes one byte) at a time. They differ from block ciphers, which operate on fixed-size blocks of data.

- **Key Expansion**: Unlike block ciphers, stream ciphers do not operate on fixed blocks of plaintext. Instead, a short key is expanded into a longer key stream that is as long as the message. This key stream is generated using a pseudo-random number generator (PRNG) or a similar mechanism.
- **Encryption Process**: The generated key stream is combined with the plaintext, typically using the bitwise exclusive-or (XOR) operation, to produce the ciphertext. Mathematically, this can be expressed as:

  $$E_{IV,k,p} = G(IV, k) \oplus p$$

  where $E$ is the encryption process, $G$ is the key stream generator, $IV$ is the initialization vector, $k$ is the key, and $p$ is the plaintext.
- **Initialization Vector (IV)**: Stream ciphers often use an IV along with the key to produce the key stream. The IV ensures that the same key produces different key streams for different sessions or messages, thereby ensuring that the same plaintext does not result in the same ciphertext across sessions. The IV is typically sent along with the ciphertext to the receiver, as it is required for decryption but doesn't need to be kept secret.
- **Importance of IV**: Without the use of an IV, encrypting two different messages with the same key would produce key streams $ks$ that could lead to vulnerabilities. For instance, given ciphertexts $c1 = p1 \oplus ks$ and $c2 = p2 \oplus ks$, an attacker could compute:

  $$c1 \oplus c2 = (p1 \oplus ks) \oplus (p2 \oplus ks) = p1 \oplus p2$$

  This would reveal information about the XOR of the two plaintexts, which is a significant security risk. The use of an IV ensures that the key streams are different for different encryptions, mitigating this risk.

In conclusion, stream ciphers offer flexibility in encrypting data of varying lengths, but care must be taken to ensure the key stream is unique for each encryption to maintain security.

## Comparison: Block Ciphers vs. Stream Ciphers

Block ciphers and stream ciphers are both symmetric encryption algorithms, but they have distinct characteristics and are suited for different use cases. Here is a comparison between the two:

- **Performance for Short Messages**: Stream ciphers can be slower than block ciphers for short messages. This is primarily due to the initialization steps in stream ciphers to produce the key stream. In contrast, block ciphers, with their fixed block size, are more efficient for these shorter lengths.
- **Performance for Long Messages**: For longer messages, stream ciphers tend to have better performance. Once the key stream is generated, encryption is a simple matter of XORing the key stream with the plaintext, which can be very fast.

- **Offline Key Stream Generation**: One advantage of stream ciphers is that the key stream can be generated offline, ahead of time. When encryption needs to be performed, the message can simply be XORed with the pre-generated key stream, allowing for potentially faster real-time encryption.
- **Trend Towards Lightweight Block Ciphers**: There is a growing trend in the design and analysis of lightweight block ciphers. These ciphers are designed to require fewer resources, making them suitable for applications with constraints on area, battery power, and other resources.

While both block and stream ciphers have their strengths, the choice between them largely depends on the specific requirements of the application and the nature of the data being encrypted.

## Structure of Block Ciphers

Block ciphers are cryptographic algorithms that encrypt data in fixed-size blocks, typically using a symmetric key. They are designed to provide both confusion and diffusion, ensuring that the relationship between the plaintext, ciphertext, and key is complex and non-obvious.

- **Round Functions**: Block ciphers operate using multiple rounds of encryption. Each round applies a series of transformations to the input data, increasing the complexity of the encryption.
- **Round Output**: The output of each round is determined by the round key and the output of the previous round. For the first round, the input is the plaintext, while the output of the last round is the ciphertext.
- **Key Schedule**: A key schedule algorithm is used to derive round keys from the main encryption key. This ensures that each round uses a different key, enhancing security.
- **Substitution and Permutation**: Block ciphers use a combination of substitution (non-linear transformation) and permutation (linear transformation) operations in their round functions. This combination ensures both confusion (making the relationship between the plaintext and ciphertext complex) and diffusion (ensuring that changes to the plaintext spread out in the ciphertext).

### Substitution-Permutation Network (SPN) Based Algorithm Structure

The Substitution-Permutation Network (SPN) is a common structure for block ciphers. It involves multiple rounds of substituting input bits with different bits (using S-boxes) and then permuting or rearranging those bits (using a linear layer). The process can be visualized as:

```
Plaintext
    |
XOR with 1st round key
    |
S-box substitutions
    |
Linear Layer
    |
... (repeated for multiple rounds)
    |
```

```
XOR with r-th round key
    |
Ciphertext
```

In this structure, the S-boxes provide non-linear transformations, ensuring complexity, while the linear layer ensures diffusion, spreading out changes across the ciphertext.

### Feistel Type Structure

Feistel ciphers are a specific type of block cipher structure. Their main characteristics include:

- **Division of Input**: The input to each round is divided into two halves.
- **Update and Swap**: Only one half is updated in each round, using a function often referred to as the $F$ function. At the end of the round, the two halves are swapped.
- $F$ **Function**: The $F$ function typically involves adding the round key, a non-linear transformation such as an S-box (substitution box), and a linear transformation like a permutation.

The Feistel structure ensures that even though only half of the data is transformed in each round, changes propagate through the entire block over multiple rounds.

### Data Encryption Standard (DES)

The Data Encryption Standard (DES) is a symmetric-key algorithm for the encryption of electronic data. Established in 1977 by the National Bureau of Standards (now NIST), DES has played a foundational role in the development of modern encryption techniques:

- **Origins**: DES is derived from the earlier LUCIFER cipher, designed by H. Feistel and others at IBM in 1970.
- **Structure**: It is structured as a Feistel cipher, consisting of 16 rounds of processing for each 64-bit block.
- **Key Size**: The algorithm uses a 56-bit key, providing a balance between security and performance at the time of its design.
- **Round Function Inputs**: The round function of DES takes a 32-bit input and combines it with a 48-bit round key.
- **Cryptanalysis**: DES is susceptible to differential and linear cryptanalysis, vulnerabilities that stem from its structure and key size.
- **Brute-Force Attacks**: The 56-bit key size, while substantial at its inception, is now considered small enough to be vulnerable to brute-force attacks with modern computing power.

DES's legacy in cryptography is significant, illustrating the evolution of cryptographic techniques and the ongoing challenge of maintaining security against increasing computational capabilities.

### Triple Data Encryption Standard (3DES)

Triple DES, often referred to as 3DES, is an enhancement of the original Data Encryption Standard (DES) algorithm, designed to provide stronger security through layered encryption:

- **Introduction**: 3DES was proposed in 1979 as a response to the vulnerabilities of DES, particularly its susceptibility to brute-force attacks.
- **Key Size Variants**: The algorithm supports key sizes of 112 bits (with $k1 = k3$) and 168 bits, effectively doubling and tripling the DES key size, respectively.
- **Encryption-Decryption-Encryption (E-D-E) Structure**: 3DES uses a sequence of three DES operations; encryption with $k1$, decryption with $k2$, and a final encryption with $k3$. The decryption step with $k2$ doesn't reverse the initial encryption because it uses a different key, thereby contributing to the overall encryption process.
- **Backward Compatibility with DES**: In scenarios where $k1 = k2 = k3$, 3DES simplifies to the original DES algorithm. This design allows 3DES to operate in environments where only DES is supported, ensuring backward compatibility.

By applying the DES cipher three times to each data block, 3DES significantly increases the difficulty of brute-force attacks, making it a more secure option at the time of its development.

### Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) is the result of a competition held by NIST to find a robust encryption standard for securing sensitive government information:

- **Selection as Standard**: AES was selected as the new standard algorithm for the United States in 1998, succeeding DES due to its higher security level.
- **Origin**: Originally named Rijndael, AES was designed by Vincent Rijmen and Joan Daemen, who submitted it to the NIST competition.
- **Block Length**: It operates on 128-bit blocks, ensuring compatibility with a wide range of applications and platforms.
- **Key Sizes**: AES supports multiple key lengths of 128, 192, and 256 bits, accommodating various security requirements and computational constraints.
- **Number of Rounds**: The algorithm's rounds depend on the key length; 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys.
- **FIPS Standardization**: AES was standardized as FIPS 197 in November 2001, reinforcing its credibility and adoption in cryptographic applications.
- **8-bit Operations**: All operations in AES are performed on 8-bit bytes. This byte-oriented approach enhances the algorithm's performance across diverse computing platforms, from hardware implementations to software execution.

AES's adoption is widespread, becoming the de facto encryption technique for various industries due to its strength and efficiency.

### Stream Ciphers

Stream ciphers are a type of symmetric encryption where plaintext is combined with a pseudo-random cipher digit stream (key stream). Each plaintext digit is encrypted one at a time with the corresponding digit of the key stream, to give a digit of the ciphertext stream.

- **Key Length**: Typically, stream ciphers use shorter key lengths compared to block ciphers, which can be advantageous in resource-constrained environments.
- **Key Stream Generation**: A smaller key can be expanded to produce the required number of key bits, creating a key stream that can be as long as the plaintext.
- **Initialization Vectors (IV)**: To maintain security, different IVs are used for each plaintext. This ensures that the same plaintext will produce different ciphertexts each time it is encrypted.

$$\begin{aligned} &\text{For encryption:} && F(k, IV, p) = G(k, IV) \oplus p = c, \\ &\text{For decryption:} && E(k, IV, c) = G(k, IV) \oplus c = p, \\ &\text{where} && G : \text{key stream generator.} \end{aligned}$$

Despite the practicality of stream ciphers in certain applications, they are not considered perfectly secure. The key stream must never be reused, as this can lead to vulnerabilities.

### Trivium

Trivium is a noteworthy example of a stream cipher that emphasizes simplicity and speed, particularly in hardware implementations:

- **Standardization**: It is recognized as part of the ISO/IEC 29192-3 standard, reflecting its acceptance in international cryptographic practices.
- **Origins**: Developed under the eSTREAM project, Trivium was designed to be a lightweight cipher suitable for a wide range of applications.
- **Designers**: The cipher was created by Christophe De Cannière and Bart Preneel, who are known for their contributions to cryptographic algorithms.
- **Internal State and Registers**: Trivium operates with a 288-bit internal state and consists of three distinct shift registers that interact during the cipher's execution.
- **Operation**: In each round of the cipher, a bit is shifted into each register, and simultaneously, a single output bit ($z_i$) is produced, contributing to the keystream.
- **Initialization**: The key and IV are loaded into the registers, followed by 1152 initialization rounds to ensure diffusion and avoid any simple relations between the key, IV, and the beginning of the output keystream.

The design of Trivium allows for efficient implementation while maintaining a high level of security, making it a practical choice for systems with limited computational resources.