**Advanced Encryption Standard (AES)**

* Advanced Encryption Standard (AES) is a highly trusted **encryption algorithm** used to secure data by converting it into an unreadable format without the proper key.
* It is developed by the National Institute of Standards and Technology (NIST) in 2001.
* It is is widely used today as it is much stronger than [DES](https://www.geeksforgeeks.org/computer-networks/data-encryption-standard-des-set-1/)and triple DES despite being harder to implement.
* **AES encryption** uses various **key lengths** (128, 192, or 256 bits) to provide strong protection against unauthorized access.
* This **data security** measure is efficient and widely implemented in securing **internet communication**, protecting **sensitive data**, and encrypting files.
* AES, a cornerstone of modern cryptography, is recognized globally for its ability to keep information safe from cyber threats.
* AES is a [Block Cipher](https://www.geeksforgeeks.org/ethical-hacking/block-cipher-modes-of-operation/).
* The key size can be 128/192/256 bits.
* Encrypts data in blocks of 128 bits each.
* It takes 128 bits as input and outputs 128 bits of encrypted cipher text.
* AES relies on the substitution-permutation network principle, which is performed using a series of linked operations that involve replacing and shuffling the input data.

## **Working of The Cipher**

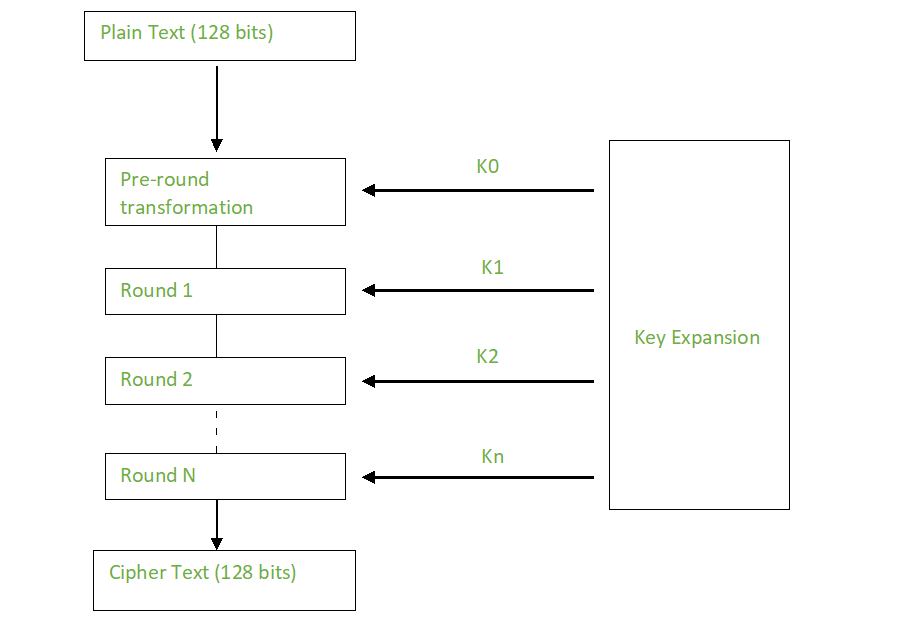
AES performs operations on bytes of data rather than in bits. Since the block size is 128 bits, the cipher processes 128 bits (or 16 bytes) of the input data at a time.

The number of rounds depends on the key length as follows :

| **N (Number of Rounds)** | **Key Size (in bits)** |
| --- | --- |
| 10 | 128 |
| 12 | 192 |
| 14 | 256 |

### **Creation of Round Keys**

A Key Schedule algorithm calculates all the round keys from the key. So the initial key is used to create many different round keys which will be used in the corresponding round of the encryption.



## **Encryption**

AES considers each block as a 16-byte (4 byte x 4 byte = 128 ) grid in a column-major arrangement.

***[ b0 | b4 | b8 | b12 |******| b1 | b5 | b9 | b13 |******| b2 | b6 | b10| b14 |******| b3 | b7 | b11| b15 ]***

**Each round comprises of 4 steps :**

* SubBytes
* ShiftRows
* MixColumns
* Add Round Key

**Step1. Sub Bytes**

This step implements the substitution.

In this step, each byte is substituted by another byte. It is performed using a lookup table also called the [S-box](https://www.geeksforgeeks.org/computer-networks/what-is-s-box-substitution/). This substitution is done in a way that a byte is never substituted by itself and also not substituted by another byte which is a compliment of the current byte. The result of this step is a 16-byte (4 x 4 ) matrix like before.

The next two steps implement the permutation.

**Step2. Shift Rows**

This step is just as it sounds. Each row is shifted a particular number of times.

* The first row is not shifted
* The second row is shifted once to the left.
* The third row is shifted twice to the left.
* The fourth row is shifted thrice to the left.

(A left circular shift is performed.)

***[ b0 | b1 | b2 | b3 ] [ b0 | b1 | b2 | b3 ]******| b4 | b5 | b6 | b7 | -> | b5 | b6 | b7 | b4 |******| b8 | b9 | b10 | b11 | | b10 | b11 | b8 | b9 |******[ b12 | b13 | b14 | b15 ] [ b15 | b12 | b13 | b14 ]***

**Step 3: Mix Columns**

This step is a matrix multiplication. Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result.

**This step is skipped in the last round.**

***[ c0 ] [ 2 3 1 1 ] [ b0 ]******| c1 | = | 1 2 3 1 | | b1 |******| c2 | | 1 1 2 3 | | b2 |******[ c3 ] [ 3 1 1 2 ] [ b3 ]***

**Step 4: Add Round Keys**

* Now the resultant output of the previous stage is XOR-ed with the corresponding round key. Here, the 16 bytes are not considered as a grid but just as 128 bits of data.
* After all these rounds 128 bits of encrypted data are given back as output. This process is repeated until all the data to be encrypted undergoes this process.

## **Decryption**

The stages in the rounds can be easily undone as these stages have an opposite to it which when performed reverts the changes. Each 128 blocks goes through the 10,12 or 14 rounds depending on the key size.

The stages of each round of decryption are as follows :

* Add round key
* Inverse MixColumns
* ShiftRows
* Inverse SubByte

### **Inverse MixColumns**

* This step is similar to the Mix Columns step in encryption but differs in the matrix used to carry out the operation.
* Mix Columns Operation each column is mixed independent of the other.
* Matrix multiplication is used. The output of this step is the matrix multiplication of the old values and a

constant matrix

***[b0] = [ 14 11 13 9] [ c0 ]******[b1]=[ 9 14 11 13 ] [ c1 ]******[b2] =[ 13 9 14 11] [ c2 ]******[ b3 ]=[ 11 13 9 14 ] [ c3 ]***

### **Inverse SubBytes**

* Inverse S-box is used as a lookup table and using which the bytes are substituted during decryption.
* Function Substitute performs a byte substitution on each byte of the input word. For this purpose, it uses an S-box.

## **Applications of AES**

AES is widely used in many applications which require secure data storage and transmission. Some common use cases include:

* **Wireless security:** AES is used in securing wireless networks, such as [Wi-Fi networks](https://www.geeksforgeeks.org/computer-networks/what-is-wi-fiwireless-fidelity/), to ensure data confidentiality and prevent unauthorized access.
* **Database Encryption:** AES can be applied to encrypt sensitive data stored in databases. This helps protect personal information, financial records, and other confidential data from unauthorized access in case of a data breach.
* **Secure communications:** AES is widely used in protocols such as internet communications, email, instant messaging, and voice/video calls. It ensures that the data remains confidential.
* **Data storage:** AES is used to encrypt sensitive data stored on hard drives, [USB drives](https://www.geeksforgeeks.org/ethical-hacking/what-are-the-security-risks-of-usb-drives/), and other storage media, protecting it from unauthorized access in case of loss or theft.
* **Virtual Private Networks (VPNs):** AES is commonly used in VPN protocols to secure the communication between a user's device and a remote server. It ensures that data sent and received through the [VPN](https://www.geeksforgeeks.org/computer-networks/what-is-vpn-how-it-works-types-of-vpn/) remains private and cannot be deciphered by eavesdroppers.
* **Secure Storage of Passwords:** AES encryption is commonly employed to store passwords securely. Instead of storing plaintext passwords, the encrypted version is stored. This adds an extra layer of security and protects user credentials in case of unauthorized access to the storage.
* **File and Disk Encryption:** [AES](https://www.geeksforgeeks.org/computer-networks/aes-full-form/) is used to encrypt files and folders on computers, external storage devices, and cloud storage. It protects sensitive data stored on devices or during data transfer to prevent unauthorized access.

### **Applications of DES**

* **Triple DES (3DES)**: A more secure variant of DES, [3-DES](https://www.geeksforgeeks.org/computer-networks/triple-des-3des/) applies DES encryption three times sequentially. It’s still used in legacy systems.
* **Financial Transactions**: DES was once used for securing financial transactions, but it has largely been replaced by AES.
* **Legacy Systems** : Some older systems still rely on DES for compatibility reasons.

## **Difference Between AES and DES**

| **AES** | **DES** |
| --- | --- |
| AES stands for [Advanced Encryption Standard](https://www.geeksforgeeks.org/computer-networks/advanced-encryption-standard-aes/) | DES stands for [Data Encryption Standard](https://www.geeksforgeeks.org/computer-networks/data-encryption-standard-des-set-1/) |
| The date of creation is 2001. | The date of creation is 1977. |
| Byte-Oriented. | Bit-Oriented. |
| Key length can be 128-bits, 192-bits, and 256-bits. | The key length is 56 bits in DES. |
| Number of rounds depends on key length: 10(128-bits), 12(192-bits), or 14(256-bits) | DES involves 16 rounds of identical operations |
| The structure is based on a substitution-permutation network. | The structure is based on a [Feistel](https://www.geeksforgeeks.org/python/feistel-cipher/) network. |
| The design rationale for AES is open. | The design rationale for DES is closed. |
| The selection process for this is secret but accepted for open public comment. | The selection process for this is secret. |
| AES is more secure than the DES cipher and is the de facto world standard. | DES can be broken easily as it has known vulnerabilities. 3DES(Triple DES) is a variation of DES which is secure than the usual DES. |
| The rounds in AES are: Byte Substitution, Shift Row, Mix Column and Key Addition | The rounds in DES are: Expansion, XOR operation with round key, Substitution and Permutation |
| AES can encrypt 128 bits of plaintext. | DES can encrypt 64 bits of plaintext. |
| It can generate Ciphertext of 128, 192, 256 bits. | It generates Ciphertext of 64 bits. |
| AES cipher is derived from an aside-channel square cipher. | DES cipher is derived from Lucifer cipher. |
| AES was designed by Vincent Rijmen and Joan Daemen. | DES was designed by IBM. |
| No known [crypt-analytical attacks](https://www.geeksforgeeks.org/computer-networks/cryptanalysis-and-types-of-attacks/) against AES but side channel attacks against AES implementations possible. Biclique attacks have better complexity than brute force but still ineffective. | Known attacks against DES include Brute-force, Linear crypt-analysis, and Differential crypt-analysis. |
| It is faster than DES. | It is slower than AES. |
| It is flexible. | It is not flexible. |
| It is efficient with both hardware and software. | It is efficient only with hardware. |

# RC5 Encryption Algorithm

* RC5 is a symmetric key block encryption algorithm designed by Ron Rivest in 1994.
* It is notable for being simple, fast (on account of using only primitive computer operations like XOR, shift, etc.) and consumes less memory.
* RC5 is a block cipher and addresses two word blocks at a time. Depending on input plain text block size, number of rounds and key size, various instances of RC5 can be defined and each instance is denoted as RC5-w/r/b where w=word size in bits, r=number of rounds and b=key size in bytes.

Allowed values are:

| **Parameter** | **Possible Value** |
| --- | --- |
| block/word size (bits) | 16, 32, 64 |
| Number of Rounds | 0 - 255 |
| Key Size (bytes) | 0 - 255 |

**Note -** Since at a time, RC5 uses 2 word blocks, the plain text block size can be 32, 64 or 128 bits. Notation used in the algorithm:

| **Symbol** | **Operation** |
| --- | --- |
| x <<< y | Cyclic left shift of x by y bits |
| + | Two's complement addition of words where addition is modulo 2w2*w* |
| ^ | Bit wise Exclusive-OR |

# How RC5 encryption algorithm works

**RC5**is a [symmetric](https://how.dev/answers/symmetric-vs-asymmetric-encryption)key algorithm for block [encryption](https://how.dev/answers/what-is-encryption)designed by Ron Rivest. It is suitable for both hardware and software implementations due to the following characteristics:

* **Simplicity:** The algorithm only uses primitive computer operations, such as addition, subtraction, bitwise XOR, and circular shifts. This makes it easy to implement and analyze.
* **Flexibility:** It allows a variable number of rounds and bit size of the key.
* **Low memory utilization:** It can be implemented on devices with limited memory.

### Parameterization

RC5 is a parameterized, word-oriented algorithm. This means it is a block cipher with a two-word input (plaintext) and a two-word output (ciphertext) block size. The parameters are detailed as follows:

### Word size: *w*

This is the word size in bits. RC5 has two w*w* bit blocks, so the input and output blocks are each 2w2*w* bits long. For example, if w*w* is 3232 bit, then the input block will be 6464 bit long.

### Number of rounds: r

This is the number of rounds and determines the trade-off between speed and security, where greater rounds imply higher security but lower speed. r*r* also influences the size t*t* of the expanded key table S*S* that is derived from the secret key. The formula for t*t* is as follows:

t=2(r+1)*t*=2(*r*+1)

### Number of bytes in secret key: b

This is the number of 88-bit bytes in the user-provided secret key, K*K*, and has the allowable range from 0−255 0−255 bytes.

The table below summarizes the parameters:

## Parameters of the RC5 encyrption algorithm

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Explanation** | **Accepted values** |
| w | Word size in bits. | 16, 32, 64 |
| r | Number of rounds. | 0–255 |
| b | Number of octets (8-bit bytes) in the secret key K. | 0–255 |

Given these parameters, the notation for the algorithm is as follows:

            RC5−w/r/b*RC*5−*w*/*r*/*b*

The nominal version of this is RC5−32/12/16*RC*5−32/12/16, which implies there are two 3232 bit word inputs and outputs, 1212 rounds, and a 1616 byte (128128 bit) key.

### **Primitive operations**

RC5 uses only three primitive operations and their inverses. These are:

* **Addition:** This refers to the two's complement addition of words and is denoted by ++. The inverse of this is subtraction, denoted by −−.
* **Bitwise XOR:** This is the bitwise exclusive-OR of words and is denoted by ⊕⊕.
* **Left rotation:** This is the cyclic left rotation of words, denoted by x<<<y*x*<<<*y*, where x*x* is the word and y*y* is the number of bits to be shifted. The inverse is cyclic right rotation, represented by x>>>y*x*>>>*y*.

The three components of the RC5 encryption algorithm:

* Key expansion algorithm
* Encryption
* Decryption

# RSA Algorithm in Cryptography

* RSA(**R**ivest-**S**hamir-**A**dleman) Algorithm is an **asymmetric**or **public-key cryptography**algorithm which means it works on two different keys: **Public Key**and **Private Key**.
* The Public Key is used for **encryption**and is known to everyone, while the Private Key is used for **decryption**and must be kept secret by the receiver. RSA Algorithm is named after Ron **R**ivest, Adi **S**hamir and Leonard **A**dleman, who published the algorithm in 1977.

**Example of Asymmetric Cryptography:**

If Person **A** wants to send a message securely to Person **B**:

* Person A **encrypts**the message using Person B's **Public**Key.
* Person B **decrypts**the message using their **Private**Key.

### **RSA Algorithm**

*RSA Algorithm is based on* ***factorization*** *of large number and* ***modular arithmetic*** *for encrypting and decrypting data. It consists of three main stages:*

1. ***Key Generation:*** *Creating Public and Private Keys*
2. ***Encryption:*** *Sender encrypts the data using Public Key to get* ***cipher text****.*
3. ***Decryption:*** *Decrypting the* ***cipher text*** *using Private Key to get the original data.*

**1. Key Generation**

* Choose two large prime numbers, say **p** and **q**. These prime numbers should be kept secret.
* Calculate the product of primes, **n = p \* q**. This product is part of the public as well as the private key.
* Calculate [Euler Totient Function](https://www.geeksforgeeks.org/dsa/eulers-totient-function/)**Φ(n)**as **Φ(n)** =**Φ(p \* q) = Φ(p) \* Φ(q) = (p - 1) \* (q - 1).**
* Choose encryption exponent **e**, such that
  + 1 < e < Φ(n), and
  + gcd(e, Φ(n)) = 1, that is e should be co-prime with Φ(n).
* Calculate decryption exponent **d,**such that
  + **(d \* e) ≡ 1 mod Φ(n)**, that is d is [modular multiplicative inverse](https://www.geeksforgeeks.org/dsa/multiplicative-inverse-under-modulo-m/) of **e**mod Φ(n). Some common methods to calculate multiplicative inverse are: [Extended Euclidean Algorithm](https://www.geeksforgeeks.org/dsa/euclidean-algorithms-basic-and-extended/), [Fermat's Little Theorem](https://www.geeksforgeeks.org/dsa/fermats-little-theorem/), etc.
  + We can have multiple values of d satisfying **(d \* e) ≡ 1 mod Φ(n)** but it does not matter which value we choose as all of them are valid keys and will result into same message on decryption.

Finally, the **Public Key = (n, e)** and the **Private Key = (n, d)**.

**2. Encryption**

To encrypt a message **M**, it is first converted to numerical representation using ASCII and other encoding schemes. Now, use the public key (n, e) to encrypt the message and get the cipher text using the formula:

***C = Me mod n****, where C is the Cipher text and e and n are parts of public key.*

**3. Decryption**

To decrypt the cipher text **C**, use the private key (n, d) and get the original data using the formula:

***M = Cd mod n,*** *where M is the message and d and n are parts of private key.*

### Advantages

* **Security:**RSA algorithm is considered to be very secure and is widely used for secure data transmission.
* **Public-key cryptography:**RSA algorithm is a public-key cryptography algorithm, which means that it uses two different keys for encryption and decryption. The public key is used to encrypt the data, while the private key is used to decrypt the data.
* **Key exchange:**RSA algorithm can be used for secure key exchange, which means that two parties can exchange a secret key without actually sending the key over the network.
* **Digital signatures:**RSA algorithm can be used for digital signatures, which means that a sender can sign a message using their private key, and the receiver can verify the signature using the sender's public key.
* **Widely used:** Online banking, e-commerce, and secure communications are just a few fields and applications where the RSA algorithm is extensively developed.

### Disadvantages

* **Slow processing speed:**RSA algorithm is slower than other encryption algorithms, especially when dealing with large amounts of data.
* **Large key size:**RSA algorithm requires large key sizes to be secure, which means that it requires more computational resources and storage space.
* **Vulnerability to side-channel attacks:** RSA algorithm is vulnerable to side-channel attacks, which means an attacker can use information leaked through side channels such as power consumption, electromagnetic radiation, and timing analysis to extract the private key.
* **Limited use in some applications:** RSA algorithm is not suitable for some applications, such as those that require constant encryption and decryption of large amounts of data, due to its slow processing speed.
* **Complexity:** The RSA algorithm is a sophisticated mathematical technique that some individuals may find challenging to comprehend and use.
* **Key Management:** The secure administration of the private key is necessary for the RSA algorithm, although in some cases this can be difficult.
* **Vulnerability to Quantum Computing:** Quantum computers have the ability to attack the RSA algorithm, potentially decrypting the data.

### **RSA Example**

The RSA algorithm is the most widely used [Asymmetric Encryption](https://www.practicalnetworking.net/series/cryptography/asymmetric-encryption/) algorithm deployed to date.

The acronym is derived from the last names of the three mathematicians who created it in 1977:  Ron **R**ivest, Adi **S**hamir, Leonard **A**dleman.

In order to understand the algorithm, there are a few terms we have to define:

* **Prime**– A number is said to be Prime if it is only divisible by 1 and itself.  Such as: 2, 3, 5, 7, 11, 13, etc.
* **Factor** – A factor is a number you can multiple to get another number.  For example, the factors of 12 are 1, 2, 3, 4, 6, and 12.
* **Semi-Prime** – A number is Semi Prime if its only factors are prime (excluding 1 and itself). For example:  
  **12** is *not* semi-prime — one of its factors is 6, which is not prime.  
  **21** *is* semi-prime — the factors of 21 are 1, **3**, **7**, 21.  If we exclude 1 and 21, we are left with 3 and 7, both of which are Prime.  
  *(Hint: Anytime you multiply two Prime numbers, the result is always Semi Prime)*
* **Modulos** – This is a fancy way of simply asking for a [remainder](https://en.wikipedia.org/wiki/Remainder).  If presented with the problem 12 MOD 5, we simply are asking for the remainder when dividing 12 by 5, which results in 2.

### RSA Key Generation

The heart of Asymmetric Encryption lies in finding two mathematically linked values which can serve as our Public and Private keys.  As such, the bulk of the work lies in the generation of such keys.

To acquire such keys, there are five steps:

1. **Select two Prime Numbers:  P and Q**

This really is as easy as it sounds.  Select two prime numbers to begin the key generation.  For the purpose of our example, we will use the numbers **7** and **19**, and we will refer to them as **P** and **Q**.

1. **Calculate the Product: (P\*Q)**

We then simply multiply our two prime numbers together to calculate the product:

7 \* 19 = **133**

We will refer to this number as **N**.   **Calculate the Totient of N: (P-1)\*(Q-1)**

To calculate the Totient of a Semi-Prime number, calculate P-1 times Q-1.

Applied to our example, we would calculate:

(7-1)\*(19-1) = 6 \* 18 = **108**

We will refer to this as **T** moving forward.

1. **Select a Public Key**

The Public Key is a value which must match three requirements:

* It must be Prime
* It must be less than the Totient
* It must NOT be a factor of the Totient

Let us see if we can get by with the number 3:  3 is indeed Prime, 3 is indeed less than 108, but regrettably 3 is a factor of 108, so we can not use it.  Can you find another number that would work?  Here is a hint, there are multiple values that would satisfy all three requirements.

For the sake of our example, we will select **29** as our **Public Key**, and we will refer to it as **E** going forward.

1. **Select a Private Key**

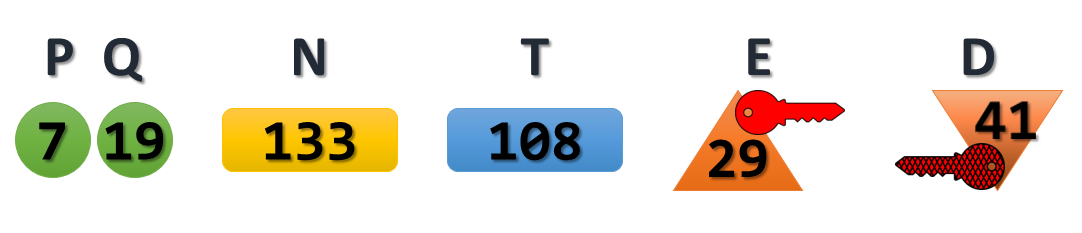
Finally, with what we have calculated so far, we can select our Private Key (which we will call **D**).The Private Key only has to match one requirement:  The Product of the Public Key and the Private Key when divided by the Totient, must result in a remainder of 1.  Or, to put it simply, the following formula must be true:

(D\*E) MOD T = 1

There are a few values that would work for the Private Key as well.  But again, for the sake of our example, we will select **41**.  To test it against our formula, we could calculate:

(41\*29) MOD 108

And there you have it, we walked through each of these five steps and ended up with the following values:



Now we simply pick a value to be used as our Plaintext message, and we can see if Asymmetric encryption really works the way they say it does.

For our example, we will go ahead and use **99**as our Plaintext message.

### Message Encryption

Using the keys we generated in the example above, we run through the Encryption process.  Recall, that with Asymmetric Encryption, we are [encrypting with the Public Key, and decrypting with the Private Key](https://www.practicalnetworking.net/series/cryptography/using-asymmetric-keys#asym-encryption).

The formula to Encrypt with RSA keys is:  **C**ipher Text = **M^E MOD N**

If we plug that into a calculator, we get:

99^29 MOD 133 = **92**

The result of **92** is our Cipher Text.  This is the value that would get sent across the wire, which only the owner of the correlating Private Key would be able to decrypt and extract the original message.  Our key pair was 29 (public) and 41 (private).  So lets see if we really can extract the original message, using our Private Key:

The formula to Decrypt with RSA keys is:  Original **M**essage = **M^D MOD N**

If we plug that into a calculator, we get:

92^41 MOD 133 = **99**

### RSA Algorithm Example

* Choose p = 3 and q = 11
* Compute n = p \* q = 3 \* 11 = 33
* Compute φ(n) = (p - 1) \* (q - 1) = 2 \* 10 = 20
* Choose e such that 1 < e < φ(n) and e
* and φ (n) are coprime. Let e = 7
* Compute a value for d such that (d \* e) MOD φ(n) = 1. One solution is d = 3 [(3 \* 7) MOD 20 = 1]
* Public key is (e, n) => (7, 33)
* Private key is (d, n) => (3, 33)
* The encryption of *m = 2* is *c = 27 MOD 33 = 29*
* The decryption of *c = 29* is *m = 293 MOD 33 = 2*