STANDARD ACCESS CONTROL LIST

A COURSE PROJECT REPORT

By

Harshitha G(RA2011030010020)
Devshree Moghe(RA2011030010049)

Under the guidance of

J Prabakaran

In partial fulfilment for the Course

of

18CSC381T - CRYPTOGRAPHY

in NWC

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Chengalpattu District

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project report "**STANDARD ACCESS CONTROL LIST**" is the bonafide work of **Harshitha G.(RA2011030010062), Devshree Moghe(RA2011030010049)** who carried out the project work under my supervision.

SIGNATURE

J PRABAKARAN
ASSISTANT PROFESSOR
NWC
SRM Institute of Science and Technology

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr**. **C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professor Dr.Annapurani Panaiyappan, Professor and Head, Department of Networking and Communications** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our my Course project Faculty **J PRABAKARAN, ASSISTANT PROFESSOR**, **NWC,** for his/her assistance, timely suggestion and guidance throughout the duration of this course project.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

1. INTRODUCTION

Access control lists are used for controlling permissions to a computer system or computer network. They are used to filter traffic in and out of a specific device. Those devices can be network devices that act as network gateways or endpoint devices that users access directly.

On a computer system, certain users have different levels of privilege, depending on their role. For example, a user logged in as network administrator may have read, write and edit permissions for a sensitive file or other resource. By contrast, a user logged in as a guest may only have read permissions.

Access control lists can help organize traffic to improve network efficiency and to give network administrators granular control over users on their computer systems and networks. ACLs can also be used to improve network security by keeping out malicious traffic.

ABSTRACT:

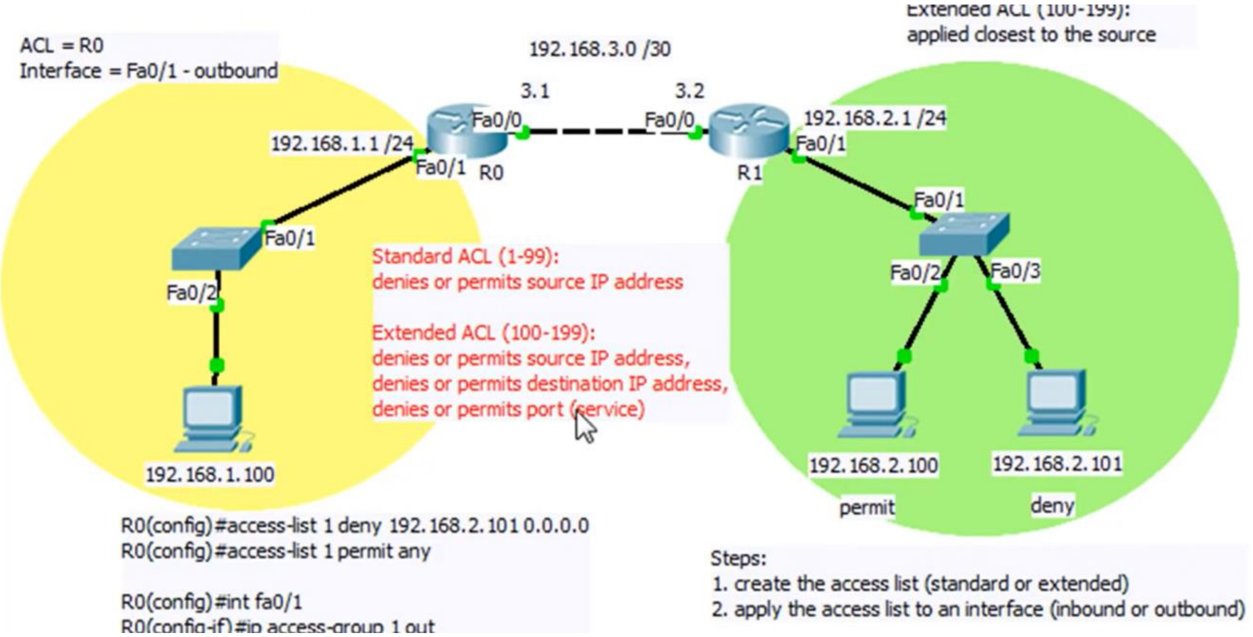
An access control list (ACL) is a list of rules that specifies which users or systems are granted or denied access to a particular object or system resource. Access control lists are also installed in routers or switches, where they act as filters, managing which traffic can access the network.

Each system resource has a security attribute that identifies its access control list. The list includes an entry for every user who can access the system. The most common privileges for a file system ACL include the ability to read a file or all the files in a directory, to write to the file or files, and to execute the file if it is an executable file or program. ACLs are also built into network interfaces and operating systems (OSes), including Linux and Windows.

2. ARCHITECTURE AND DESIGN

2.1 Network Architecture

The network architecture is as follows:



Extended ACL (100-199):
applied closest to the source

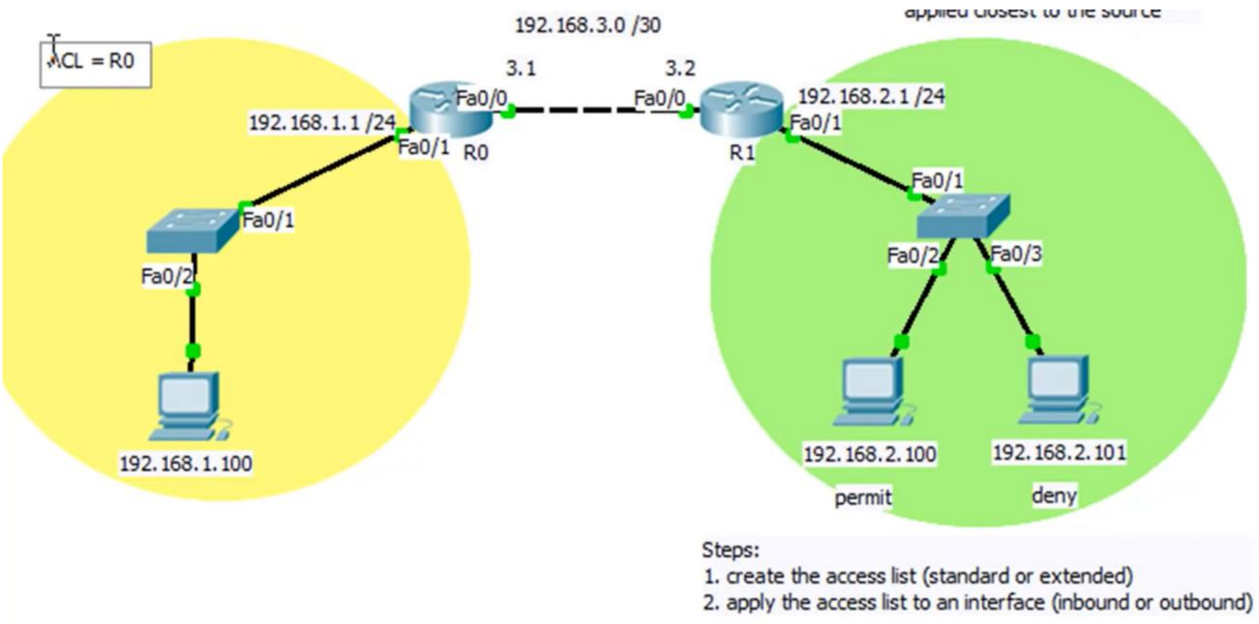
ACL = R0
Interface = Fa0/1 - outbound

Standard ACL (1-99):
denies or permits source IP address

Extended ACL (100-199):
denies or permits source IP address,
denies or permits destination IP address,
denies or permits port (service)

```
R0(config)#access-list 1 deny 192.168.2.101 0.0.0.0
R0(config)#access-list 1 permit any

R0(config)#int fa0/1
R0(config-if)#ip access-group 1 out
```

Steps:
1. create the access list (standard or extended)
2. apply the access list to an interface (inbound or outbound)



applied closest to the source

ACL = R0

Steps:
1. create the access list (standard or extended)
2. apply the access list to an interface (inbound or outbound)

3. IMPLEMENTATION

3.1 Address Table

The address table is as follows:

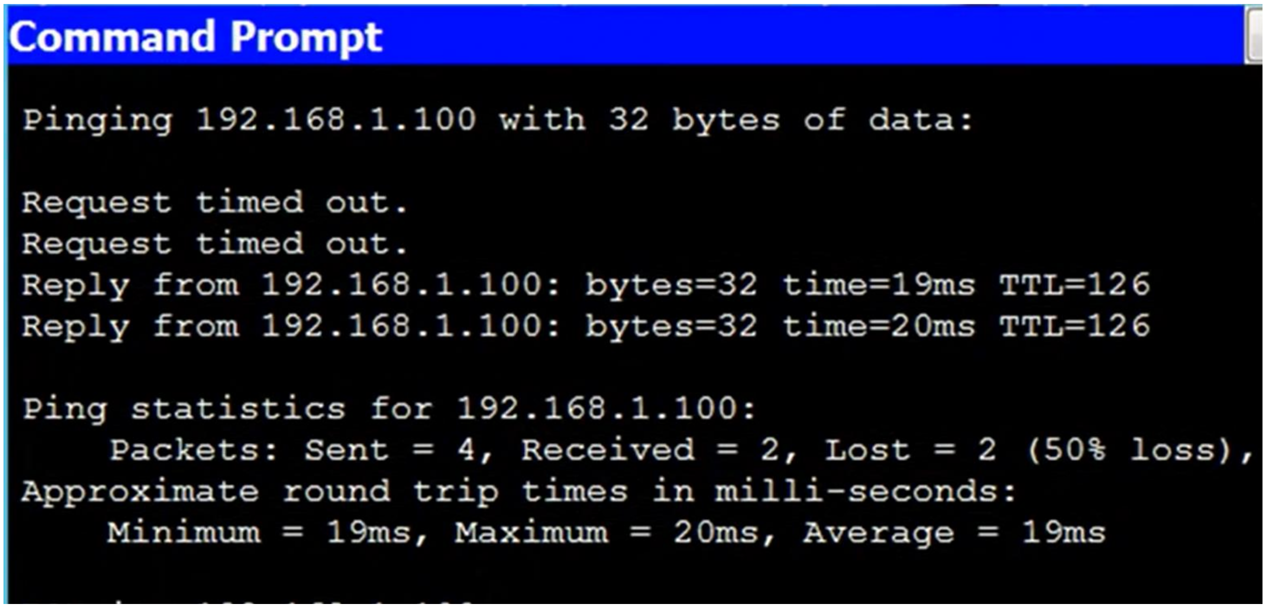
Device	**Interface**	**Address**
PC0 PC1	Fa0 Fa0	192.168.1.100 192.168.2.100
PC2	Fa0/0	192.168.2.101
PC3	Fa0	192.168.20.30
	Fa0/0	192.168.20.40
ROUTER 3	Fa0/1	192.168.1.1
ROUTER4	Fa0/1	192.168.2.1

4.RESULTS AND DISCUSSION

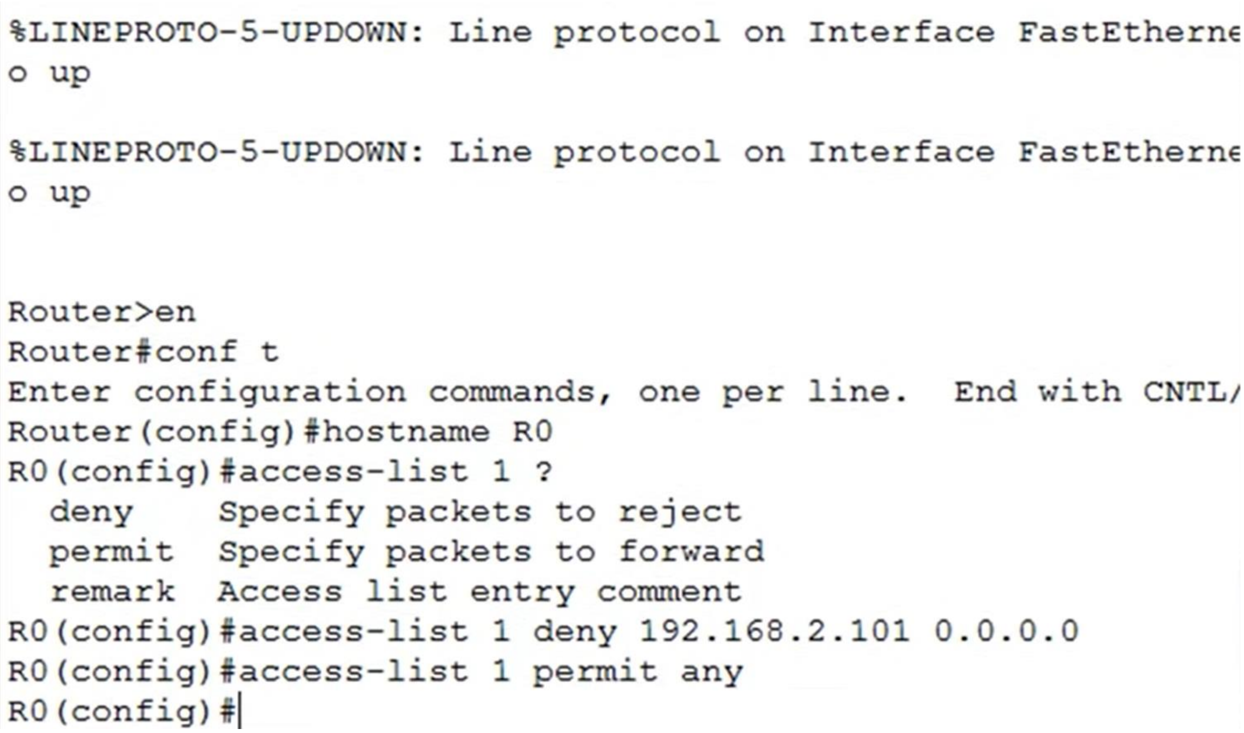
3.2 Connection Check

The network connections were checked by ping requests:

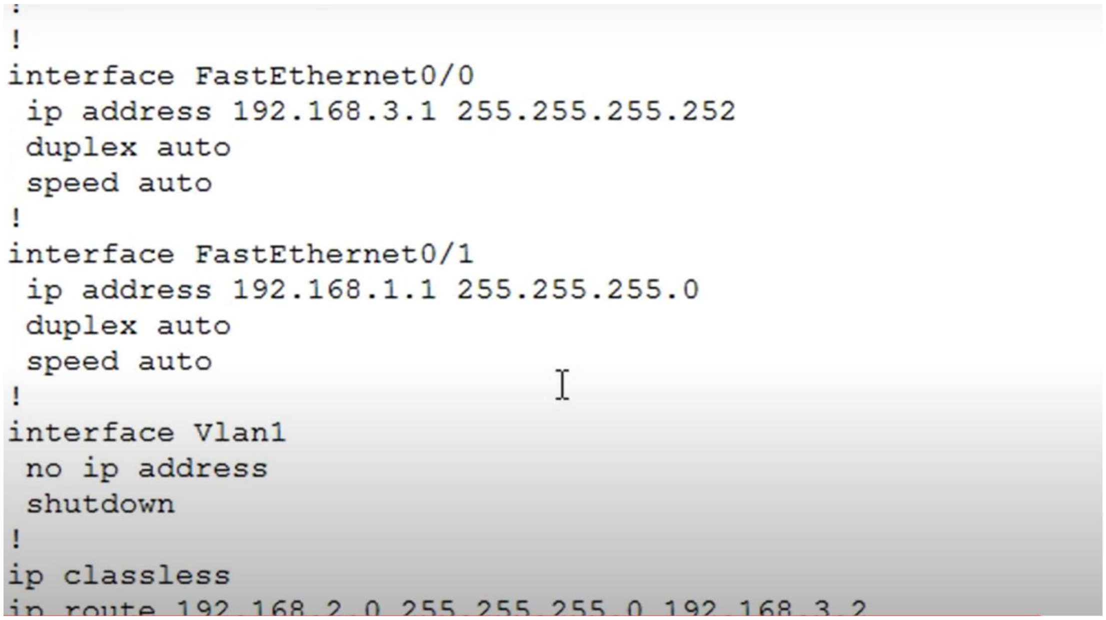
STUDENTS PC:



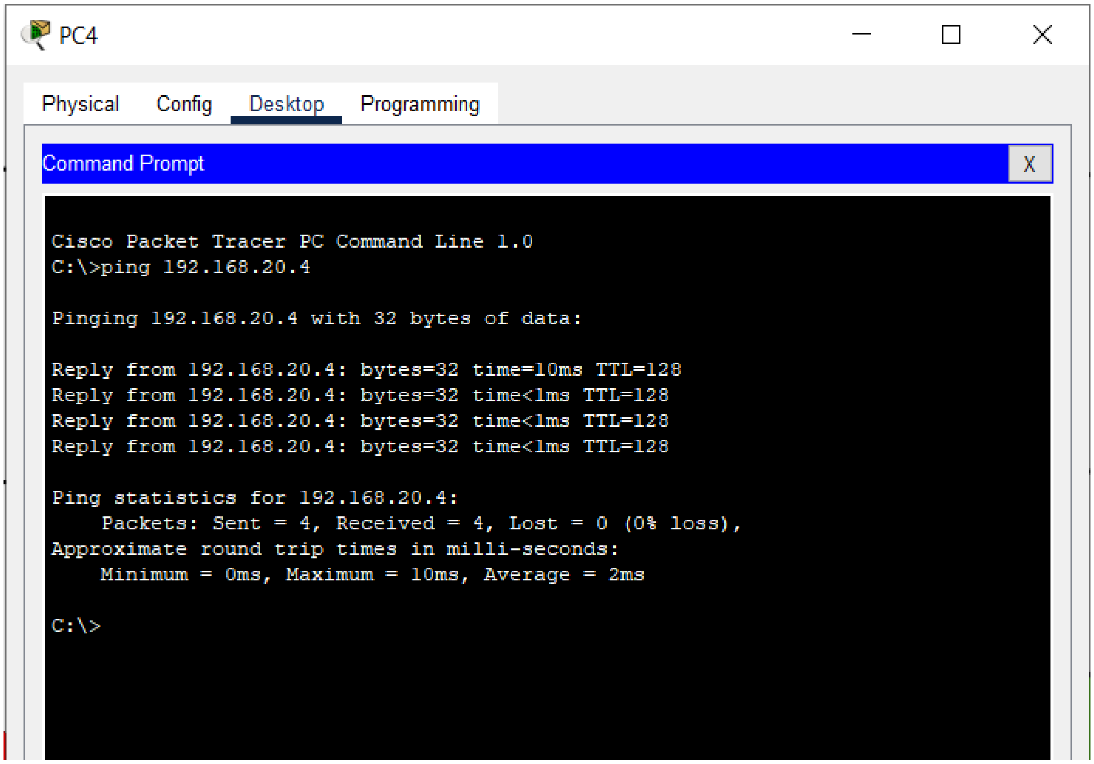
```
Pinging 192.168.1.100 with 32 bytes of data:

Request timed out.
Request timed out.
Reply from 192.168.1.100: bytes=32 time=19ms TTL=126
Reply from 192.168.1.100: bytes=32 time=20ms TTL=126

Ping statistics for 192.168.1.100:
    Packets: Sent = 4, Received = 2, Lost = 2 (50% loss),
Approximate round trip times in milli-seconds:
    Minimum = 19ms, Maximum = 20ms, Average = 19ms
```



```
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEtherne
o up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEtherne
o up


Router>en
Router#conf t
Enter configuration commands, one per line.  End with CNTL/
Router(config)#hostname R0
R0(config)#access-list 1 ?
  deny    Specify packets to reject
  permit  Specify packets to forward
  remark  Access list entry comment
R0(config)#access-list 1 deny 192.168.2.101 0.0.0.0
R0(config)#access-list 1 permit any
R0(config)#
```



```
!
!
interface FastEthernet0/0
 ip address 192.168.3.1 255.255.255.252
 duplex auto
 speed auto
!
interface FastEthernet0/1
 ip address 192.168.1.1 255.255.255.0
 duplex auto
 speed auto
!
interface Vlan1
 no ip address
 shutdown
!
ip classless
ip route 192.168.2.0 255.255.255.0 192.168.3.2
```



```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.20.4

Pinging 192.168.20.4 with 32 bytes of data:

Reply from 192.168.20.4: bytes=32 time=10ms TTL=128
Reply from 192.168.20.4: bytes=32 time<1ms TTL=128
Reply from 192.168.20.4: bytes=32 time<1ms TTL=128
Reply from 192.168.20.4: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.20.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 10ms, Average = 2ms

C:\>
```

Basic ACL implementation:

The ACL started as a very basic traffic control mechanism that provided engineers with the ability to control which traffic was allowed to enter or exit their devices; another way to think of it was as a basic firewall mechanism.

Since its original implementation, the use of the ACL has extended considerably, this article will take a look at the basic functionality of the ACL and how it can be used to perform basic access control. We'll cover advanced access control lists later, including the more advanced ways that the ACL can be configured and how it can be used outside of traffic filtering.

As stated above, the original intention of the access control lists was to provide a mechanism to enable the filtering of specific traffic into and out of a device. This section will take a look at how an ACL can be configured on a Cisco device to offer basic filtering services.

The most basic form of an ACL that is typically taught is the standard IP ACL, with this type of ACL only the source IP address is matched; the syntax of this form of ACL is shown below:

```
router(config)#access-list access-list-number {permit | deny} {any |
host host-ip-address | ip-address wildcard-mask}
```

For a standard ACL, the access-list-number is set from 1-99 or 1300-1999. The part of this type of ACL (and most other ACL's types) that throws people off is the wildcard-mask. The wildcard-mask is used to determine which specific addresses are being matched with the ACL statement. Why it is so confusing for new network engineers is because it is formatted in a way that is not “normally” seen. The wildcard-mask is the inverse of the common subnet mask; for example, if the network 192.168.1.0 255.255.255.0 matched addresses from 192.168.1.0 through 192.168.1.255 then the wildcard mask that would match these same hosts would be 0.0.0.255.

The easiest way to look at this is in binary form (255.255.255.0):

```
11111111   11111111   11111111   00000000
```

Now the wildcard mask (0.0.0.255):

00000000   00000000   00000000   11111111

While the example shown here may not be that hard to follow, it can get a little interesting when matching different subnets. For example, let's take a look at matching the network 172.16.100.64 255.255.255.192:

(255.255.255.192)

11111111   11111111   11111111   11000000

(0.0.0.63)

00000000   00000000   00000000   00111111

Hopefully these examples can help a little in understanding how to determine the correct wildcard mask to use when implementing an access control list.

ACCESS CONTROL LIST IMPLEMENTATION:

```
     router(config)#interface f0/1

     router(config-if) #ip access-group 10 out

   router(config)#ip access-list standard {access-list-name}
```

Deny:

```
   router(config)#access-list 10 deny 192.168.10.10 0.0.0.255
   router(config)#access-list 10 deny 192.168.10.20 0.0.0.255
  router(config)#access-list 10 deny 192.168.10.30 0.0.0.255
```

Permit:

```
router(config)#access-list 10 deny 192.168.20.30 0.0.0.255
router(config)#access-list 10 deny 192.168.20.40 0.0.0.255
router(config)#access-list 10 deny 192.168.20.50 0.0.0.255
```

ASYMMETRIC AND SYMMETRIC ALGORITHMS

Asymmetric cryptography, also known as public-key cryptography, is a process that uses a pair of related keys -- one public key and one private key -- to encrypt and decrypt a message and protect it from unauthorized access or use.

A public key is a cryptographic key that can be used by any person to encrypt a message so that it can only be decrypted by the intended recipient with their private key. A private key -- also known as a secret key -- is shared only with key's initiator.

When someone wants to send an encrypted message, they can pull the intended recipient's public key from a public directory and use it to encrypt the message before sending it. The recipient of the message can then decrypt the message using their related private key.

If the sender encrypts the message using their private key, the message can be decrypted only using that sender's public key, thus authenticating the sender. These encryption and decryption processes happen automatically; users do not need to physically lock and unlock the message.

Many protocols rely on asymmetric cryptography, including the transport layer security (TLS) and secure sockets layer (SSL) protocols, which make HTTPS possible.

The encryption process is also used in software programs that need to establish a secure connection over an insecure network, such as browsers over the internet, or that need to validate a digital signature.

Increased data security is the primary benefit of asymmetric cryptography. It is the most secure encryption process because users are never required to reveal or share their private keys, thus decreasing the chances of a cybercriminal discovering a user's private key during transmission.

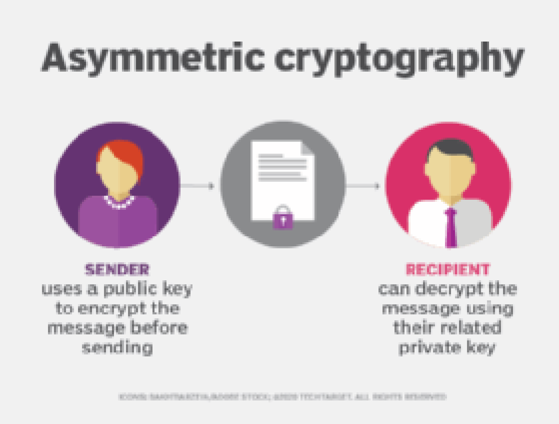
Symmetric encryption is a type of encryption where only one key (a secret key) is used to both encrypt and decrypt electronic data. The entities communicating via symmetric encryption must exchange the key so that it can be used in the decryption process. This encryption method differs from asymmetric encryption where a pair of keys - one public and one private - is used to encrypt and decrypt messages.

By using symmetric encryption algorithms, data is "scrambled" so that it can't be understood by anyone who does not possess the secret key to decrypt it. Once the intended recipient who possesses the key has the message, the algorithm reverses its action so that the message is returned to its original readable form. The secret key that the sender and recipient both use could be a specific password/code or it can be random string of letters or numbers that have been generated by a secure random number generator (RNG). For banking-grade encryption, the symmetric keys must be created using an RNG that is certified according to industry standards, such as FIPS 140-2.

Aim: To perform asymmetric and symmetric cryptographic algorithms

Procedure (for Asymmetric):

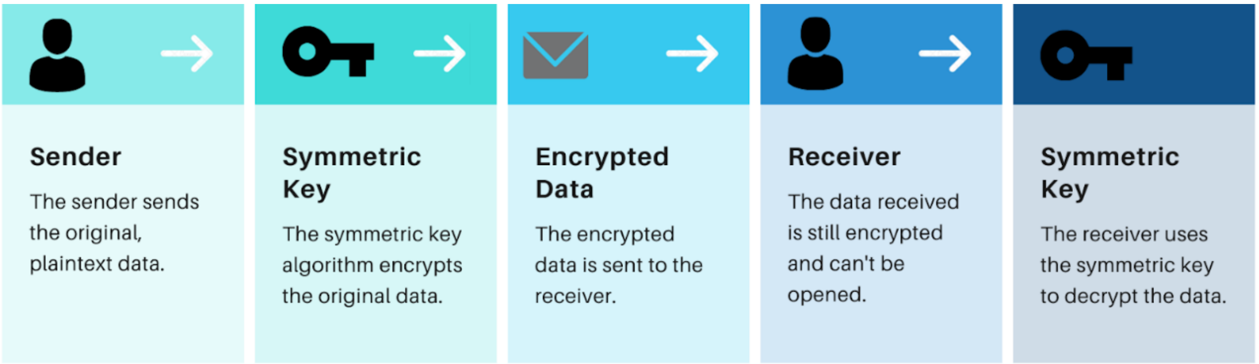
- Asymmetric encryption uses a mathematically related pair of keys for encryption and decryption: a public key and a private key. If the public key is used for encryption, then the related private key is used for decryption. If the private key is used for encryption, then the related public key is used for decryption.


Asymmetric cryptography
SENDER
uses a public key to encrypt the message before sending
RECIPIENT
can decrypt the message using their related private key

- The two participants in the asymmetric encryption workflow are the sender and the receiver.
- Each has its own pair of public and private keys.
- First, the sender obtains the receiver's public key. Next, the plaintext message is encrypted by the sender using the receiver's public key. This creates ciphertext.
- The ciphertext is sent to the receiver, who decrypts it with their private key, returning it to legible plaintext.

(Symmetric):

- In symmetric encryption, the key that encrypts a message or file is the same key that can decrypt them.
- The sender of the data uses the symmetric key algorithm to encrypt the original data and turn it into cipher text.
- The encrypted message is then sent to the receiver who uses the same symmetric key to decrypt or open the cipher text or turn it back into readable form.



Sender
The sender sends the original, plaintext data.

Symmetric Key
The symmetric key algorithm encrypts the original data.

Encrypted Data
The encrypted data is sent to the receiver.

Receiver
The data received is still encrypted and can't be opened.

Symmetric Key
The receiver uses the symmetric key to decrypt the data.

Sample Case for asymmetric encryption by using RSA:

Let us take an example of this procedure to learn the concepts. For ease of reading, it can write the example values along with the algorithm steps.

- Choose two large prime numbers P and Q
  Let P = 47, Q = 17
- Calculate N = P x Q
  We have, N = 7 x 17 = 119.
- Choose the public key (i.e., the encryption key) E such that it is not an element of (P -1) x (Q – 1)
  1. Let us find (7 - 1) x (17 -1) = 6 x 16 = 96
  2. The factors of 96 are 2, 2, 2, 2, 2, and 3 (because 96 = 2 x 2 x 2 x 2 x 2 x 3).
  3. Therefore, it can select E such that none of the factors of E is 2 and 3. We cannot choose E as 4 (because it has 2 as a factor), 15 (because it has 3 as a factor) and 6 (because it has 2 and 3 both as factors).
  4. Let us choose E as 5 (it can have been any other number that does not its factors as 2 and 3).
- Choose the private key (i.e., the decryption key) D including the following equation is true:
  (D x E) mod (P – 1) x (Q – 1) = 1
  1. Let us substitute the values of E, P, and Q in the equation.
  2. We have (D x 5) mod (7 – 1) x (17 – 1) = 1.
  3. That is, (D x 5) mod (6) x (16) = 1.
  4. That is, (D x 5) mod (96) = 1
  5. After some calculations, let us take D = 77. Then the following is true: (77 x 5) mod (96) = 385 mod 96 = 1 which is what we wanted.
- For encryption, calculate the cipher text (CT) from the plain text (PT) as follows:
  CT = PT^E mod N
  Let us assume that we want to encrypt plain text 10. Then, we have
  CT = 10^5 mod 119 = 100000 mod 119 = 40.
- Send CT as the cipher text to the receiver.

Send 40 as the cipher text to the receiver.

- For decryption, calculate the plain text (PT) from the cipher text (CT) as follows:

  $PT = CT^D \bmod N$

  It perform the following:

  $PT = CT^D\bmod N$

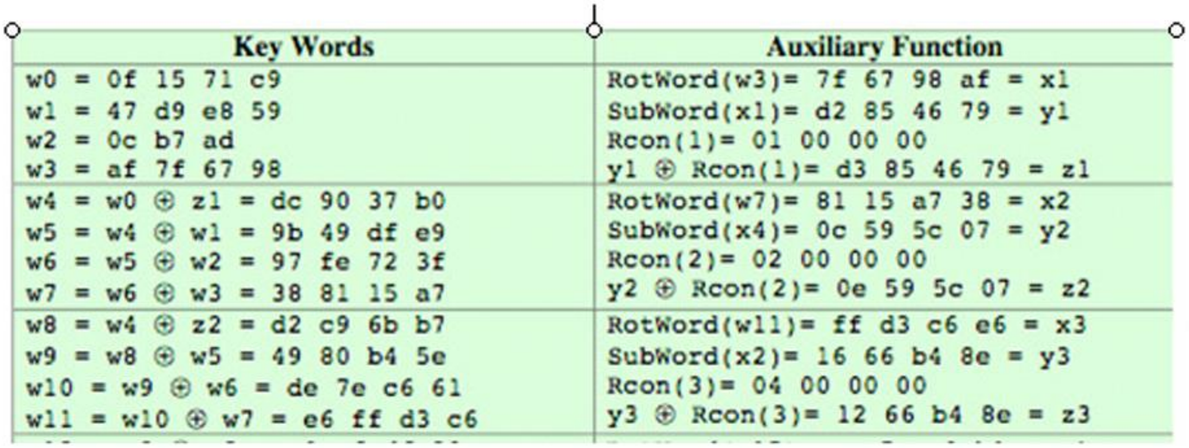
  That is,

  $PT = 40^{77}\bmod 119 = 10$, which was the original plaintext of step5.

Sample case for symmetric encryption by using AES:

- Rounds Nr = 6 + max{Nb, Nk}Nb= 32-bit words in the block Nk= 32-bit words in key
  1. Substitute Bytes.Each byte is replaced by byte indexed by row (left 4-bits) & column (right 4-bits) of a 16x16 table
  2. Shift Rows:1st row is unchanged. 2nd row does 1 byte circular shift to left 3rd row does 2 byte circular shift to left .4th row does 3 byte circular shift to left.
  3. Mix Columns:Effectively a matrix multiplication in GF(28) using prime polynomial m(x) =x8+x4+x3+x+1
- Uses arithmetic in the finite field GF(28) with irreducible polynomial m(x) = x8 + x4 + x3 + x + 1 which is (100011011) or {11B} Example: {02} • {87} mod {11B} = (1 0000 1110) mod {11B} = (1 0000 1110) (1 0001 1011) = (0001 0101)
- XOR state with 128-bits of the round key
- Use four byte words called wi. Subkey = 4 words. For AES-128: First subkey (w3,w2,w1,w0) = cipher key Other words are calculated as follows: wi=wi-1 wi-4 1. for all values of i that are not multiples of 4. For the words with indices that are a multiple of 4 (w4k): RotWord: Bytes of w4k-1 are rotated left shift (nonlinearity) 2. 3. SubWord: SubBytes fn is applied to all four bytes. (Diffusion) The result rsk is XOR'ed with w4k-4 and a round constant rconk (breaks Symmetry): w4k=rsk w4k-4 rconk For AES-192 and AES-256, the key expansion is more complex.

AES Example Key Expansion



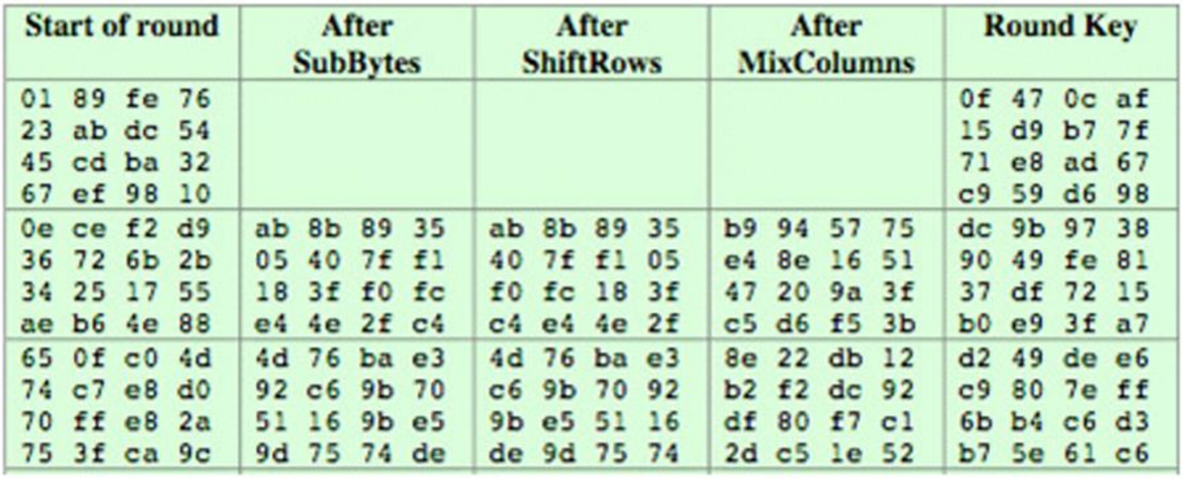
Key Words	Auxiliary Function

w0 = 0f 15 71 c9
w1 = 47 d9 e8 59
w2 = 0c b7 ad
w3 = af 7f 67 98
	RotWord(w3)= 7f 67 98 af = x1
	SubWord(x1)= d2 85 46 79 = y1
	Rcon(1)= 01 00 00 00
	y1 ⊕ Rcon(1)= d3 85 46 79 = z1

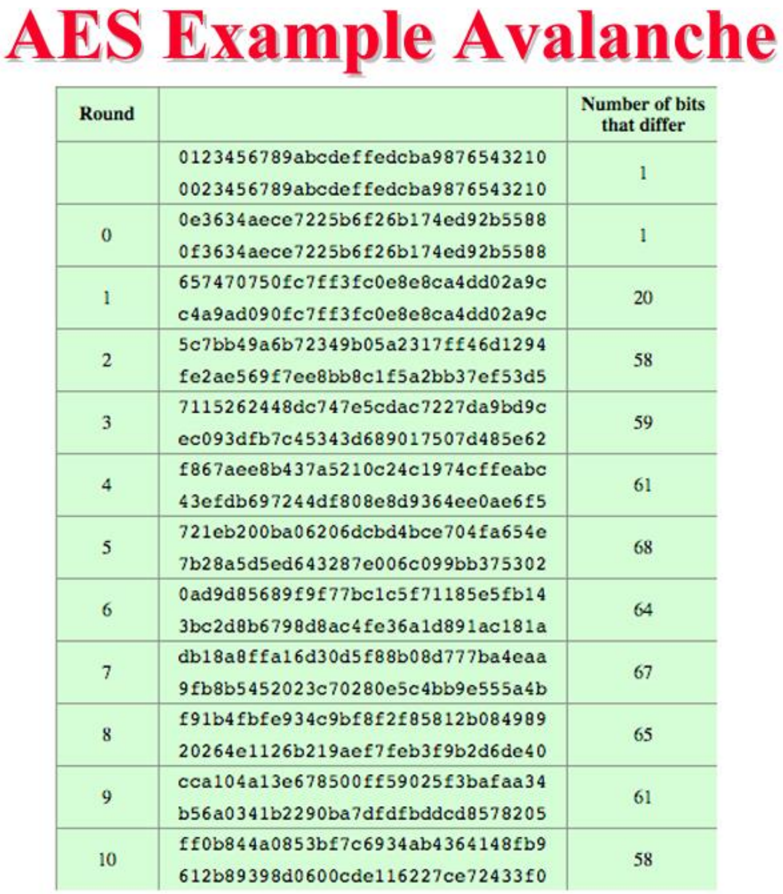
w4 = w0 ⊕ z1 = dc 90 37 b0
w5 = w4 ⊕ w1 = 9b 49 df e9
w6 = w5 ⊕ w2 = 97 fe 72 3f
w7 = w6 ⊕ w3 = 38 81 15 a7
	RotWord(w7)= 81 15 a7 38 = x2
	SubWord(x4)= 0c 59 5c 07 = y2
	Rcon(2)= 02 00 00 00
	y2 ⊕ Rcon(2)= 0e 59 5c 07 = z2

w8 = w4 ⊕ z2 = d2 c9 6b b7
w9 = w8 ⊕ w5 = 49 80 b4 5e
w10 = w9 ⊕ w6 = de 7e c6 61
w11 = w10 ⊕ w7 = e6 ff d3 c6
	RotWord(w11)= ff d3 c6 e6 = x3
	SubWord(x2)= 16 66 b4 8e = y3
	Rcon(3)= 04 00 00 00
	y3 ⊕ Rcon(3)= 12 66 b4 8e = z3

AES Example Encryption



Start of round	After SubBytes	After ShiftRows	After MixColumns	Round Key
01 89 fe 76 23 ab dc 54 45 cd ba 32 67 ef 98 10				0f 47 0c af 15 d9 b7 7f 71 e8 ad 67 c9 59 d6 98
0e ce f2 d9 36 72 6b 2b 34 25 17 55 ae b6 4e 88	ab 8b 89 35 05 40 7f f1 18 3f f0 fc e4 4e 2f c4	ab 8b 89 35 40 7f f1 05 f0 fc 18 3f c4 e4 4e 2f	b9 94 57 75 e4 8e 16 51 47 20 9a 3f c5 d6 f5 3b	dc 9b 97 38 90 49 fe 81 37 df 72 15 b0 e9 3f a7
65 0f c0 4d 74 c7 e8 d0 70 ff e8 2a 75 3f ca 9c	4d 76 ba e3 92 c6 9b 70 51 16 9b e5 9d 75 74 de	4d 76 ba e3 c6 9b 70 92 9b e5 51 16 de 9d 75 74	8e 22 db 12 b2 f2 dc 92 df 80 f7 c1 2d c5 1e 52	d2 49 de e6 c9 80 7e ff 6b b4 c6 d3 b7 5e 61 c6



AES Example Avalanche

Round		Number of bits that differ
	0123456789abcdeffedcba9876543210 0023456789abcdeffedcba9876543210	1
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c4a9ad090fc7ff3fc0e8e8ca4dd02a9c	20
2	5c7bb49a6b72349b05a2317ff46d1294 fe2ae569f7ee8bb8c1f5a2bb37ef53d5	58
3	7115262448dc747e5cdac7227da9bd9c ec093dfb7c45343d689017507d485e62	59
4	f867aee8b437a5210c24c1974cffeabc 43efdb697244df808e8d9364ee0ae6f5	61
5	721eb200ba06206dcbd4bce704fa654e 7b28a5d5ed643287e006c099bb375302	68
6	0ad9d85689f9f77bc1c5f71185e5fb14 3bc2d8b6798d8ac4fe36a1d891ac181a	64
7	db18a8ffa16d30d5f88b08d777ba4eaa 9fb8b5452023c70280e5c4bb9e555a4b	67
8	f91b4fbfe934c9bf8f2f85812b084989 20264e1126b219aef7feb3f9b2d6de40	65
9	cca104a13e678500ff59025f3bafaa34 b56a0341b2290ba7dfdfbddcd8578205	61
10	ff0b844a0853bf7c6934ab4364148fb9 612b89398d0600cde116227ce72433f0	58

- AES decryption is not identical to encryption .But each step has an inverse.

Asymmetric cryptography is typically used to authenticate data using digital signatures. A digital signature is a mathematical technique used to validate the authenticity and integrity of a message, software or digital document. It is the digital equivalent of a handwritten signature or stamped seal.

Based on asymmetric cryptography, digital signatures can provide assurances of evidence to the origin, identity and status of an electronic document, transaction or message, as well as acknowledge informed consent by the signer.

Asymmetric cryptography can also be applied to systems in which many users may need to encrypt and decrypt messages, including:

- **Encrypted email.** A public key can be used to encrypt a message and a private key can be used to decrypt it.
- **SSL/TLS.** Establishing encrypted links between websites and browsers also makes use of asymmetric encryption.

- **Cryptocurrencies.** Bitcoin and other cryptocurrencies rely on asymmetric cryptography. Users have public keys that everyone can see and private keys that are kept secret. Bitcoin uses a cryptographic algorithm to ensure only legitimate owners can spend the funds.

Uses of symmetric key algorithm:

Banking Sector. Due to the better performance and faster speed of symmetric encryption, symmetric cryptography is typically used for bulk encryption of large amounts of data. Applications of symmetric encryption in the banking sector include:

  - Payment applications, such as card transactions where PII (Personal Identifying Information) needs to be protected to prevent identity theft or fraudulent charges without huge costs of resources. This helps lower the risk involved in dealing with payment transactions on a daily basis.
  - Validations to confirm that the sender of a message is who he claims to be.

- Data at rest. Data at rest is data that is not actively moving from device to device or network-to-network such as data stored on a hard drive, laptop, flash drive, or archived/stored in some other way. Data protection at rest aims to secure inactive data stored on any device or network. While data at rest is sometimes considered to be less vulnerable than data in transit, attackers often find data at rest a more valuable target than data in motion. For protecting data at rest, enterprises can simply encrypt sensitive files prior to storing them and/or choose to encrypt the storage drive itself.
- The best way to encrypt data at rest is by whole disk or full disk encryption. Full disk encryption has several benefits compared to regular file or folder encryption, or encrypted vaults. Nearly everything including the swap space and the temporary files is encrypted. Encrypting these files is important, as they can reveal important confidential data. With a software implementation, the bootstrapping code cannot be encrypted, however. For example, BitLocker Drive Encryption leaves an unencrypted volume to boot from, while the volume containing the operating system is fully encrypted. In addition, the decision of which individual files to encrypt is not left up to users' discretion. This is important for situations in which users might not want or might forget to encrypt sensitive files.

CONCLUSION:

Hence the implementation of Access Control List and Asymmetric and Symmetric algorithms have been successfully implemented.