Des cipher:-

Des is a block chain cipher which generally encrypts data in blocks of size of 64 bits.

64 bits of plain text go as input and 64 bit of ciphertext is produed.

The same key are used for encryption and decrption and the size of the key is 56 bit.

#why the key size is 56 bit:-

Before the des processstart ,every 8th bit of the key is discarded to produce a 56 bit key. The bit positions are 8,16,24 ans so on.

The step by step working of Des:-

Des consists of 16 steps each of which are knowm as rounds

* In the first step, the 64-bit plain text block is handed over to an initial Permutation (IP) function.
* The initial permutation is performed on plain text.
* Next, the initial permutation (IP) produces two halves of the permuted block; saying Left Plain Text (LPT) and Right Plain Text (RPT).
* Now each LPT and RPT go through 16 rounds of the encryption process.
* In the end, LPT and RPT are rejoined and a Final Permutation (FP) is performed on the combined block
* The result of this process produces 64-bit ciphertext

AES:-

The AES includes three block ciphers

1:- AES-128 uses a 128 bit key length to encrypt and decrypt data

2:- AES-192 uses a 192-bit key length to encrypt and decrypt a block of messages.

3:- AES-256 uses a 256-bit key length to encrypt and decrypt a block of messages.

This cipher is generally used to encrypt classified information by organisation such as the government.

There are 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys.

A round consists of several processing steps that include substitution, transposition and mixing of the input [plaintext](https://www.techtarget.com/searchsecurity/definition/plaintext) to transform it into the final output of [ciphertext](https://www.techtarget.com/whatis/definition/ciphertext).

Deffile hellman key exchange

The **Diffie-Hellman key exchange can be explained via Merkle’s puzzles:-**

The so-called[Merkle’s Puzzles](http://www.merkle.com/1974/PuzzlesAsPublished.pdf) involve one party creating and sending a number of cryptographic puzzles to the other. These puzzles would take a moderate amount of computational resources to solve.

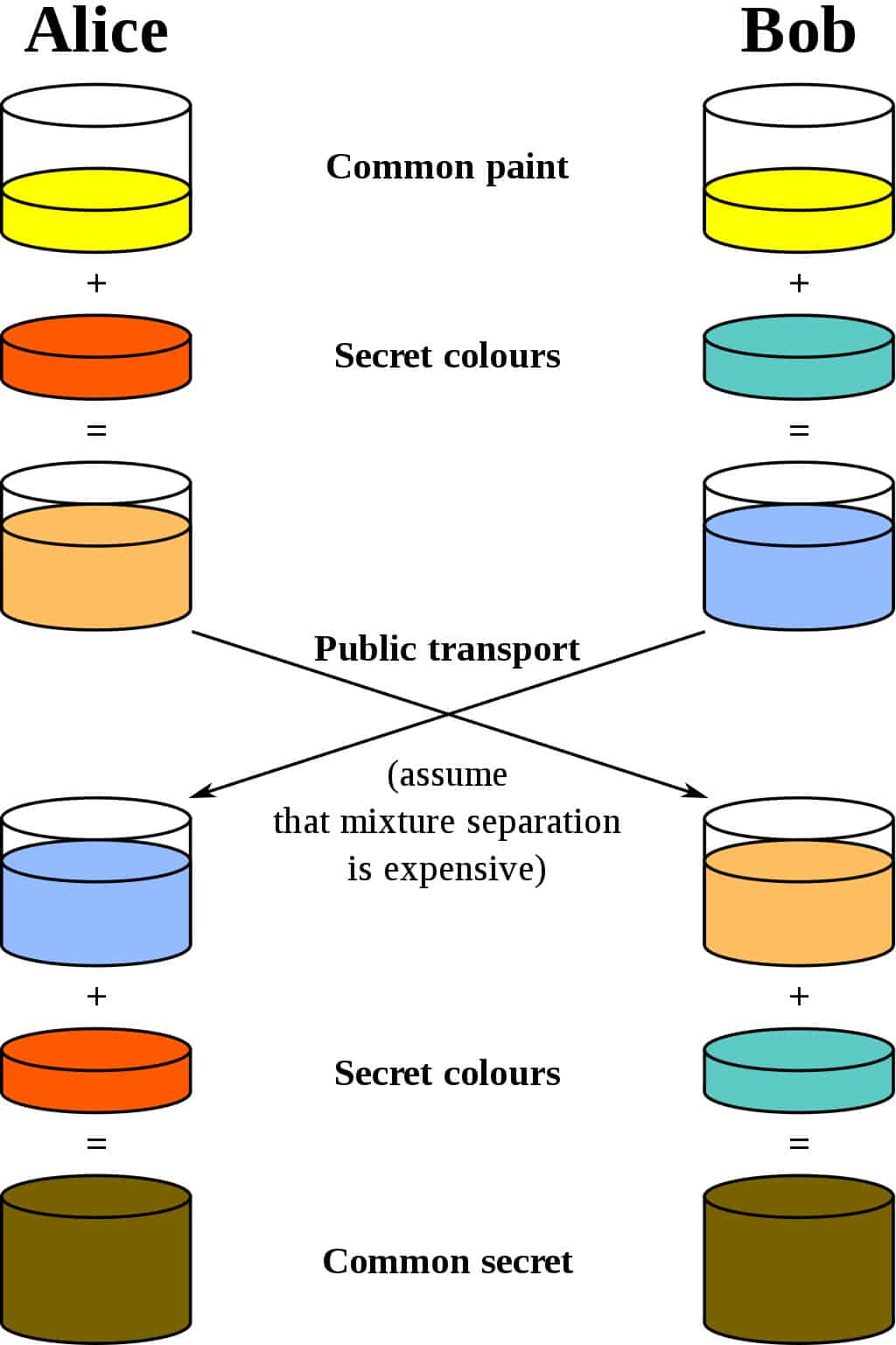
The recipient would randomly choose one puzzle to solve and then expend the necessary effort to complete it. Once the puzzle is solved, an identifier and a session key are revealed to the recipient. The recipient then transmits the identifier back to the original sender, which lets the sender know which puzzle has been solved.

Since the original sender created the puzzles, the identifier lets them know which session key the recipient discovered, and the two parties can use this key to communicate more securely. If an attacker is listening in on the interaction, they will have access to all of the puzzles, as well as the identifier that the recipient transmits back to the original sender.

The identifier doesn’t tell the attacker which session key is being used, so the best approach for decrypting the information is to solve all of the puzzles to **solve all of the puzzles to uncover the correct session key**. Since the attacker will have to solve half of the puzzles on average, it ends up being much more difficult for them to uncover the key than it is for the recipient.

How does the Diffie-Hellman key exchange work?

The best analogy for the Diffie-Hellman scheme is to think of two people mixing paint. Let’s use the cryptography standard, and say that their names are Alice and Bob. They both agree on a random color to start with. Let’s say that they send each other a message and decide on yellow as their common color, just like in the diagram below:



they set their own color. They do not tell the other party their choice. Let’s say that Alice chooses red, while Bob chooses a slightly-greenish blue.

The next step is for both Alice and Bob to mix their secret color (red for Alice, greenish-blue for Bob) with the yellow that they mutually agreed upon. According to the diagram, Alice ends up with an orangish mix, while Bob’s result is a deeper blue.

Once they have finished the mixing, they send the result to the other party. Alice receives the deeper blue, while Bob is sent the orange-colored paint.

Once they have received the mixed result from their partner, they then add their secret color to it. Alice takes the deeper blue and adds her secret red paint, while Bob adds his secret greenish-blue to the orange mix he just received.

The result? They both come out with the same color, which in this case is a disgusting brown. It may not be the kind of color that you would want to paint your living room with, but it is a shared color nonetheless. This shared color is referred to as the common secret.

**Elliptical curve cryptography (ECC)**

What is elliptical curve cryptography (ECC)?

Elliptical curve cryptography (ECC) is a [public key](https://www.techtarget.com/searchsecurity/definition/public-key) encryption technique based on elliptic curve theory that can be used to create faster, smaller and more efficient cryptographic keys.

ECC is an alternative to the Rivest-Shamir-Adleman ([RSA](https://www.techtarget.com/searchsecurity/definition/RSA)) cryptographic algorithm and is most often used for digital signatures in cryptocurrencies, such as Bitcoin and Ethereum, as well as one-way encryption of emails, data and software.

Public key cryptography systems, like ECC, use a mathematical process to merge two distinct keys and then use the output to encrypt and decrypt data. One is a public key that is known to anyone, and the other is a [private key](https://www.techtarget.com/searchsecurity/definition/private-key) that is only known by the sender and receiver of the data.

ECC generates keys through the properties of an elliptic curve equation instead of the traditional method of generation as the product of large prime numbers. From a cryptographic perspective, the points along the graph can be formulated using the following equation:

*y²=x³ + ax + bs*