User-chosen password vs. system-assigned password

What's the difference between **theoretical password space** and **effective password space**?

Theoretical Password Space: This assumes that every character is used equally often when creating a password. For example, if you have a set of letters, numbers, and symbols, each character (like 'a', 'z', '#') is just as likely to appear as any other. This gives us the maximum possible combinations.

Effective Password Space: This is the actual number of different passwords people tend to use in real life. When users pick their own passwords, they often choose predictable patterns (like common words, names, or easy-to-remember sequences). This makes the effective space much smaller than the theoretical space because many combinations are ignored.

The theoretical password space assumes that all characters appear with equal frequency. In reality, passwords contain more 'a' or 'e' characters, compared to 'z' or '#' characters. When users choose a password, the effective password space is therefore far lower than the theoretical password space.

When a system randomly assigns a password, the theoretical password space becomes equal to the effective password space.

Password entropy range:

Entropy below **20** bits - **Pin-level security**
Entropy between **20** to **60** bits - **Password-level security**
Entropy greater than **60** bits - **Crypto-level security**

Why do we need an authentication secret that offers crypto-level security?

We don't need it to authenticate to regular commercial websites, but we need it for high-stakes applications because these applications protect highly sensitive information that, if compromised, can have severe consequences.

High-stakes such as:
a. Master password for password manager
b. Password for protecting private keys
c. Admin account password

How user chosen passwords are susceptible against guessing attacks, dictionary attacks, and brute-force attacks but system-assigned passwords are not.

Guessing Attacks: When users select their own passwords, they use personally memorable things such as favorite movie character, pet name etc. That's why user-chosen passwords are easily guessable especially for someone who knows the user or has access to their public information.

Dictionary Attacks: Attackers analyze common passwords to create a password cracking dictionary. Attackers use lists of common passwords (like "password123" or "qwerty") to crack passwords quickly. Since many people use predictable or simple passwords, this method can easily find user-chosen passwords.

Brute-Force Attacks: As for brute-force, if a user-chosen password policy is not strong (like being too short or not using special characters), they become easier and faster to crack through brute force.

System-assigned passwords can protect against all these attacks. For example, if we enforce that a user has to use a system-assigned random password of twelve lowercase letters, then the theoretical entropy space = effective entropy space = 56 bits

What is the most critical disadvantage of password or “what you know” Based authentication?

Ans: **If the secret is stolen, then the person can be impersonated.**

What is the solution?

Ans: **hashing the password**

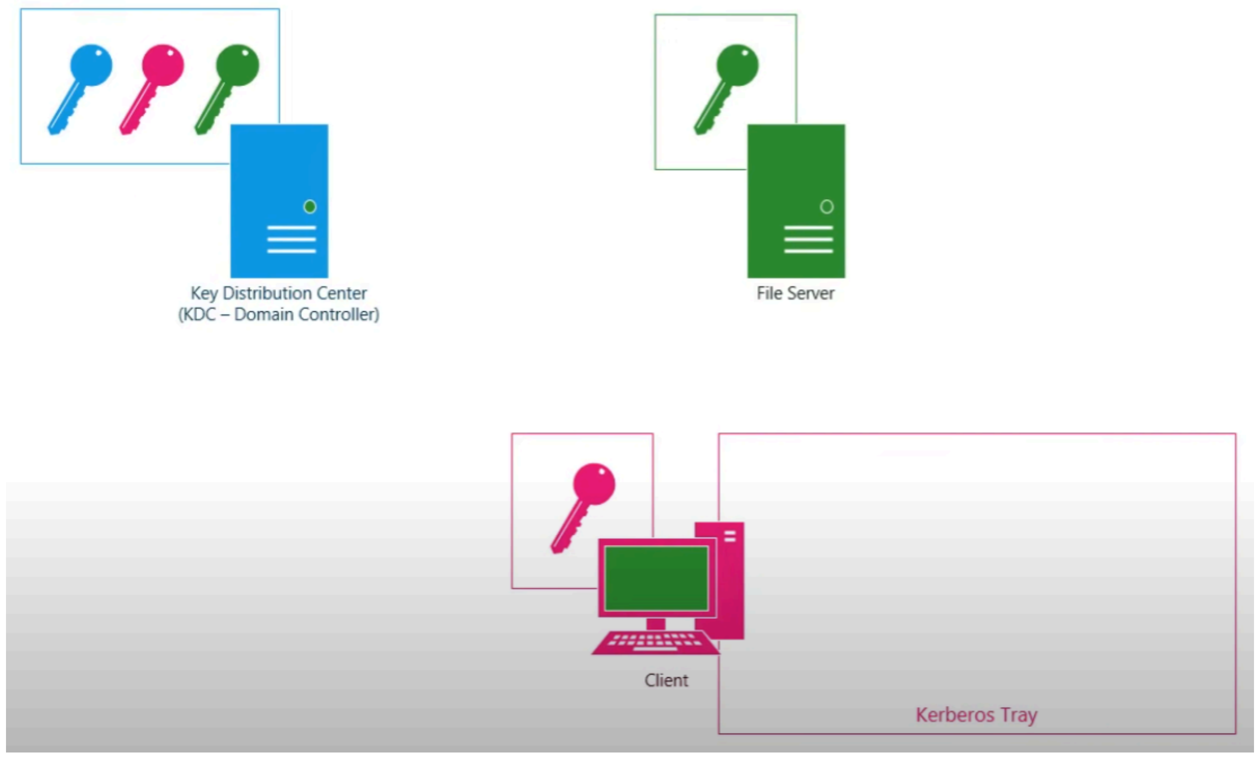
What is hashing?

Hashing is an encryption process used in cryptography to encrypt data (like a password or message) into a fixed-size string of characters, which typically appears random. Hashing is a one-way function that takes input data of any size and converts it into a fixed-size output, called a hash or hash value. The output is a unique string (like a series of letters and numbers), known as a hash.

Encryption is two-way whereas hashing is a one-way function.

Alternatively, there is another method of authentication that does not require the password to be transmitted across the network. The most popular such protocol is **Kerberos authentication protocol.**

In Greek mythology, Kerberos is a three-headed hound, which is analogically equivalent to three parties involved in a network-based authentication paradigm: authenticator, client, and service provider.



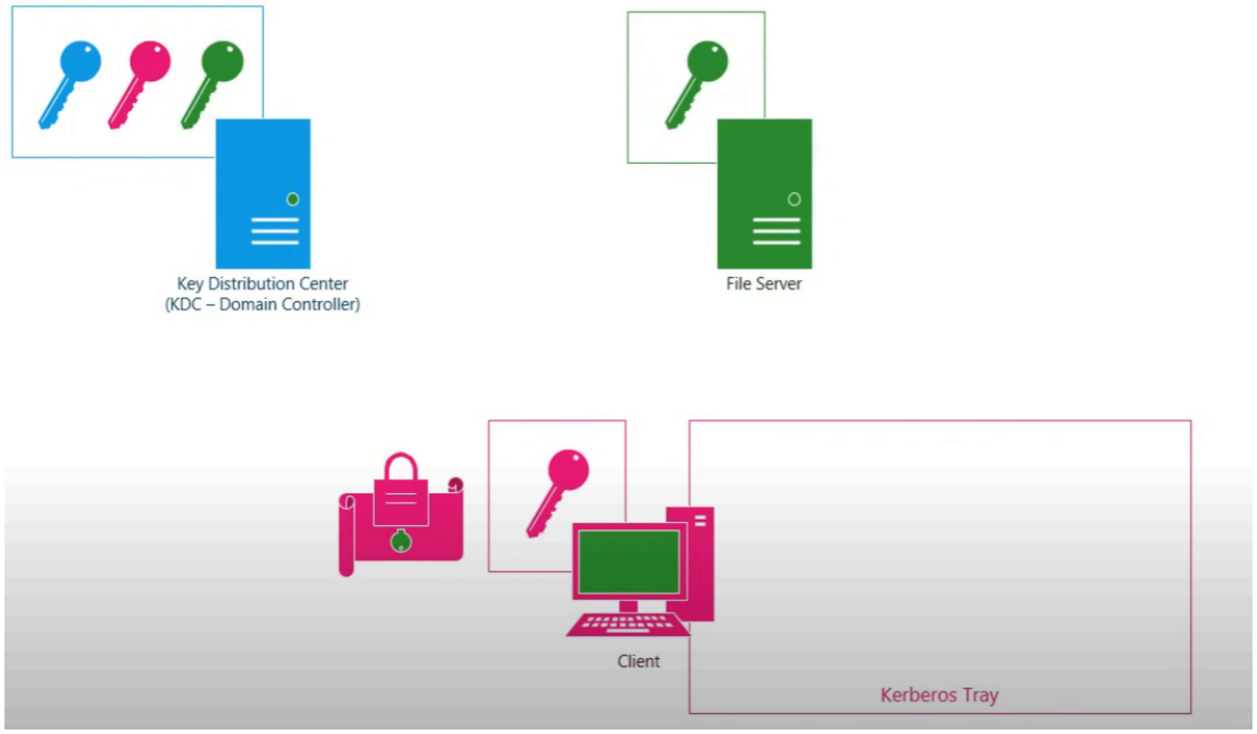
Client: The user or device trying to access a service.

Authenticator (Key Distribution Center - KDC): The trusted third party responsible for verifying identities and issuing tickets.

Service Provider: The server hosting the service the user wants to access (like email or file sharing).

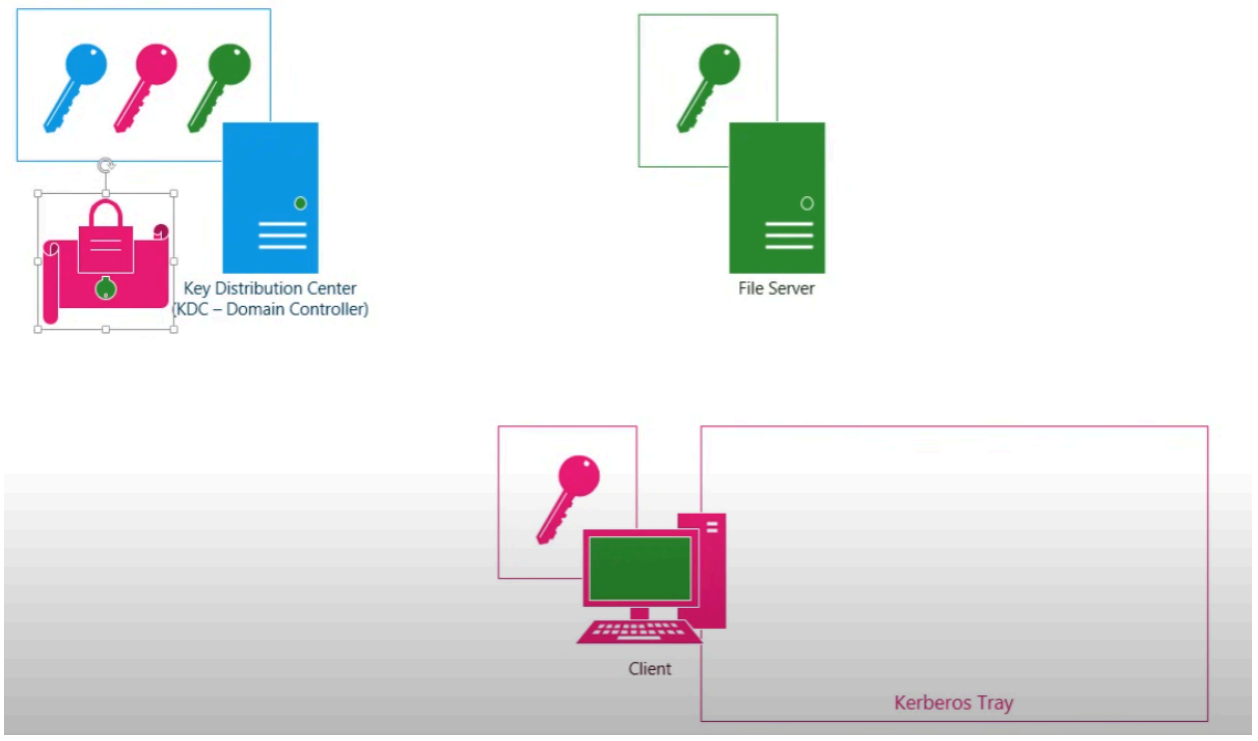
In Kerberos there is no communication between KDC and service provider. The client takes on majority of the processing burden in kerberos.

When the client wants to log in to the network, it constructs an authenticator, let's say a package that includes information about who the user is like his name, date, time and other information. These authenticators are only good for a limited time so that they are not captured by an attacker and replay back on the network. A portion of this authenticator is not encrypted for example the username so that KDC can identify the user who is trying to access and authenticate them. Now as we know in kerberos it does not require the password to be transmitted across the network so what the user will do is the user will encrypt a portion of it using the user's own password.



User encrypts the authenticator with its password.

Here, the password acts as a shared secret.

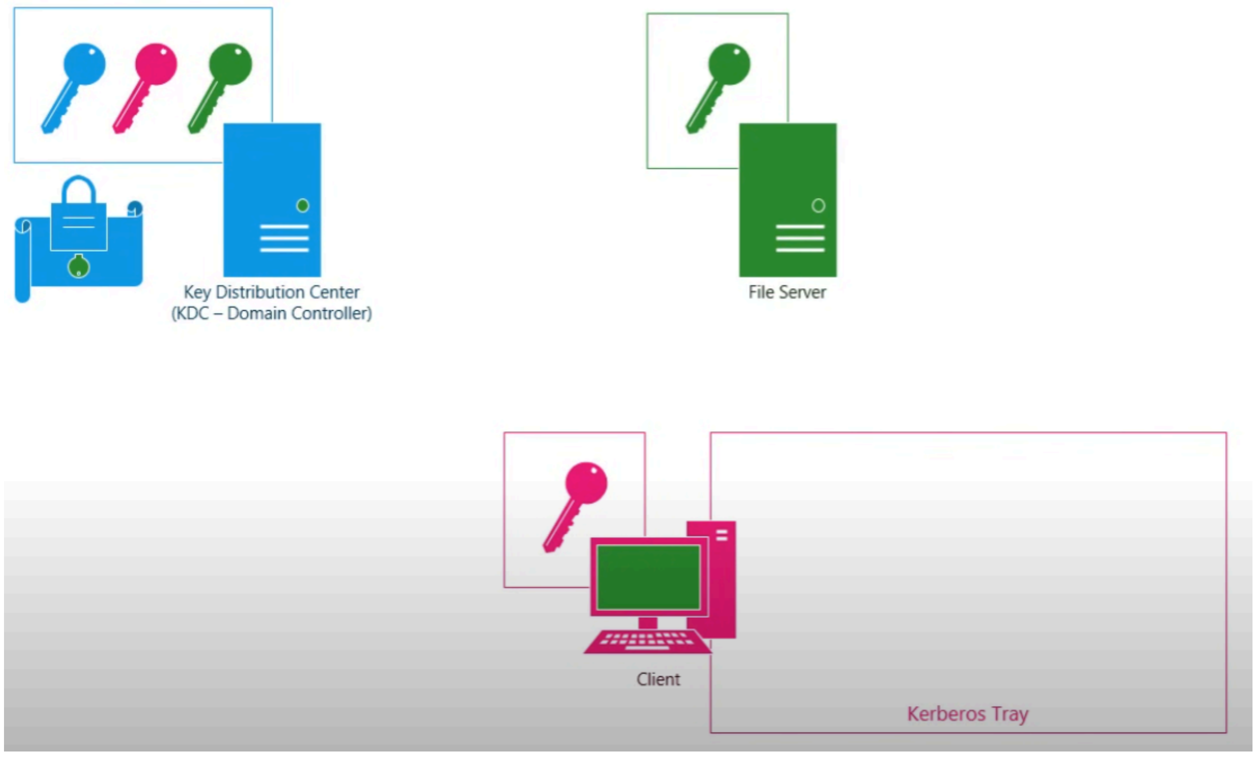


Sent to the domain controller(KDC)

After encryption, this is sent to the domain controller according to the picture above. The Domain controller can open it up and see in clear text, who the user is claiming to be. Now what the KDC will do is it will look for the password of the user in its database and try to decrypt the authenticator. If the password fails to decrypt the authenticator, that means the user is not who they are claiming to be. And if the authenticator is decrypted successfully, that means the user must be who they say they are.

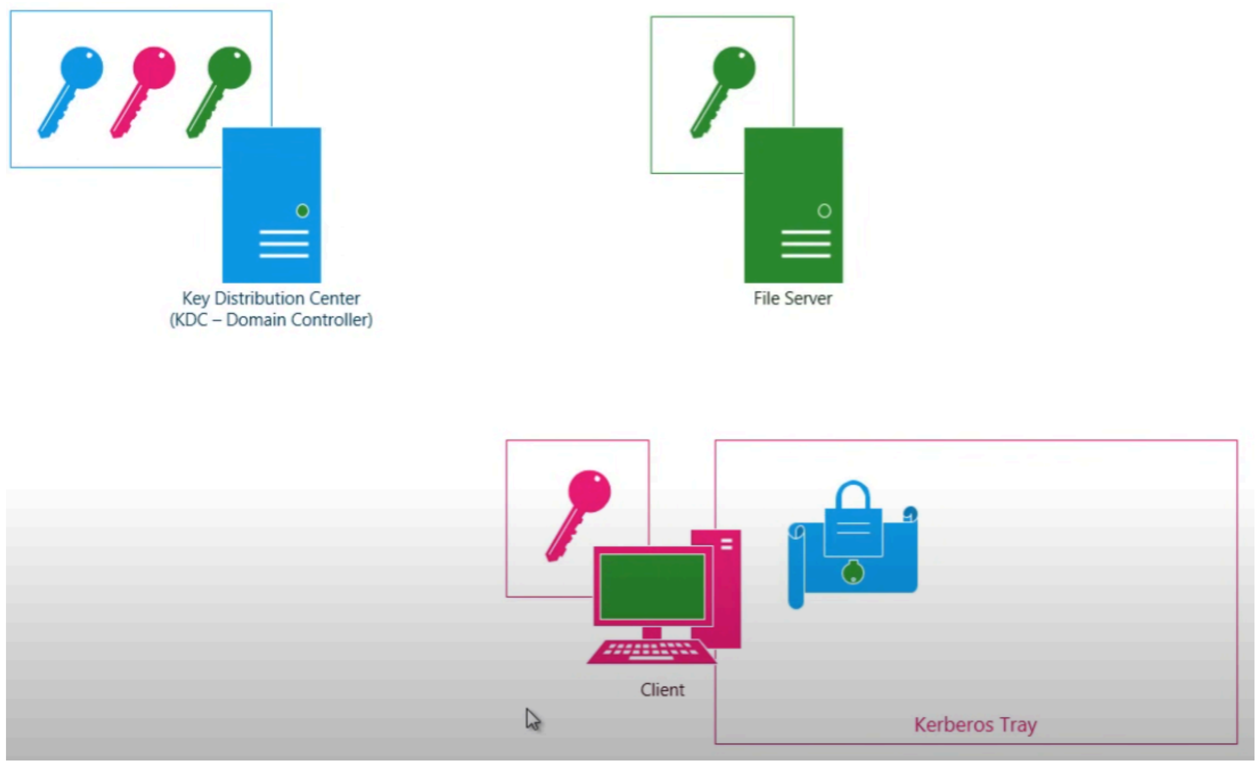
Now how does the KDC have my password? Although kerberos authentication protocol does not require the password to be transmitted across the network?

Our password is transmitted only once when we log into the network for the first time and that's the only time our password is transmitted through the network and KDC saves our password in its database.



Encrypts using KDC's own encryption key

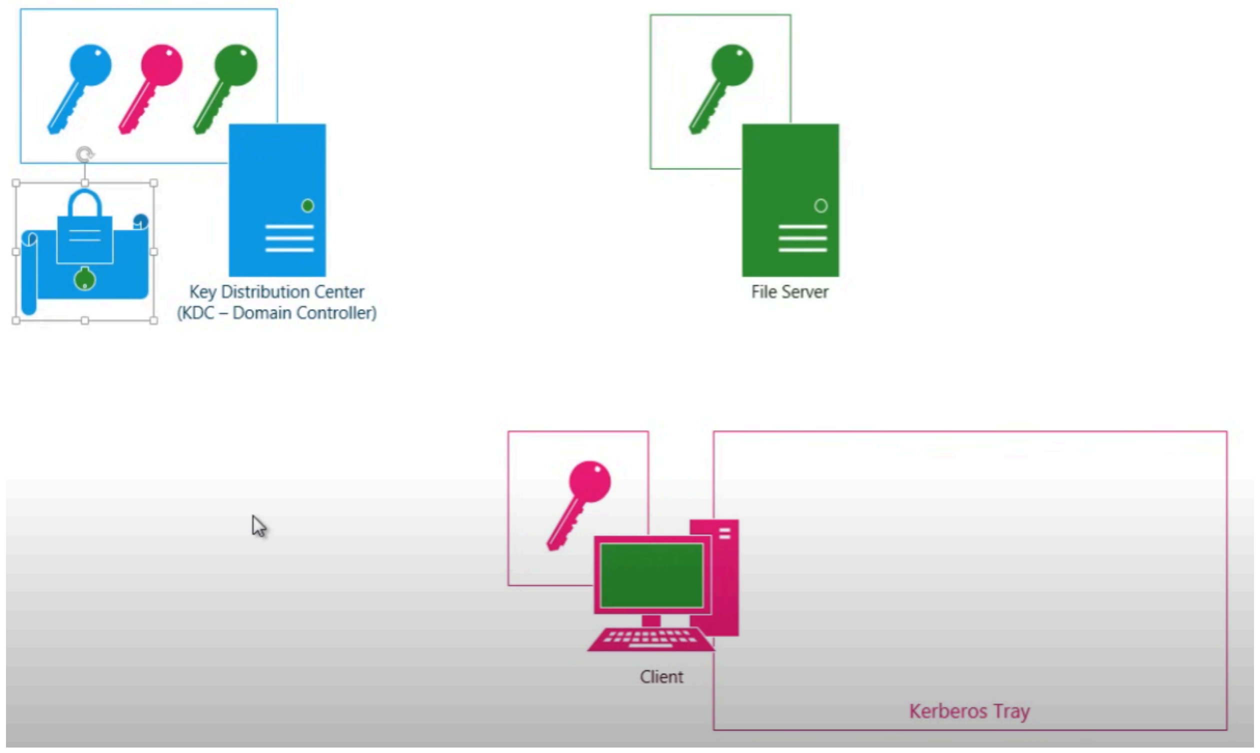
In the next step what KDC will do is, it will create a ticket granting ticket(TGT). The domain controller(KDC) has its own encryption key(Blue) which it will use to encrypt a portion of this TGT. TGT contains information about the user, who they are and transmits it back to the client and the client will store it in a special area called Kerberos Tray.



Stored in kerberos tray

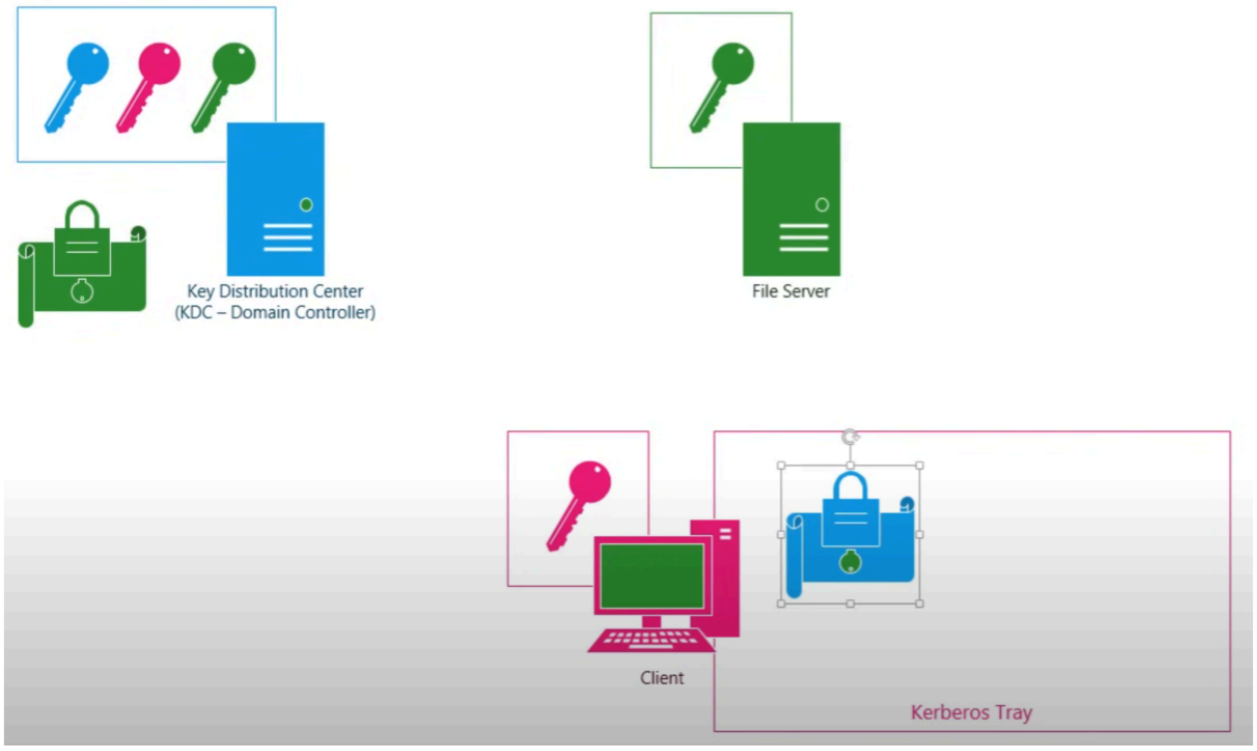
This kerberos tray is an area of memory that cannot be swapped out to disk, it always lives in the memory and if the computer crashes, this information is lost as it is not saved anywhere.

Now, the client is logged on and all ready to get something from the file server. Initially the client does not have a ticket to get access to the file server and they will go back to KDC by sending the TGT they received earlier along with a request (“i need a ticket for the file server”).

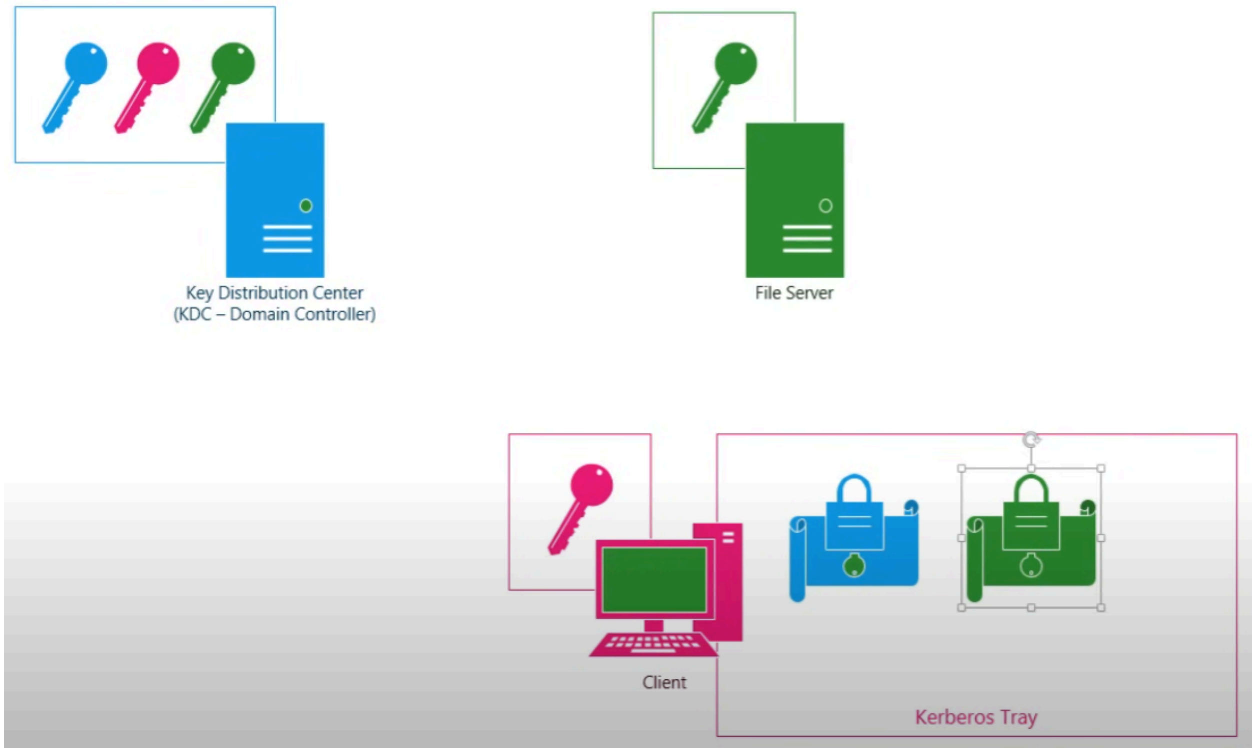


Requests for file server access

This time the KDC does not require to re-verify the client as this step has been done already. It simply uses its key(Blue) to decrypt the TGT and if it decrypts successfully then it knows that this is the information it generated itself previously. These TGT's are usually good for eight hours by default so periodically they are required to be destroyed and renewed so the KDC gets the opportunity to revalidate the clients identity. The client actually keeps a copy of this TGT in its kerberos tray and sent a copy to the domain controller requesting for the file server access and the KDC knows who the client is and thus generates a ticket. As the file server is also a domain member, the KDC has its logon password(Green key). The KDC creates a new TGT and uses the logon password of the file server to encrypt the TGT and gives it to the client who stores it in the kerberos tray.

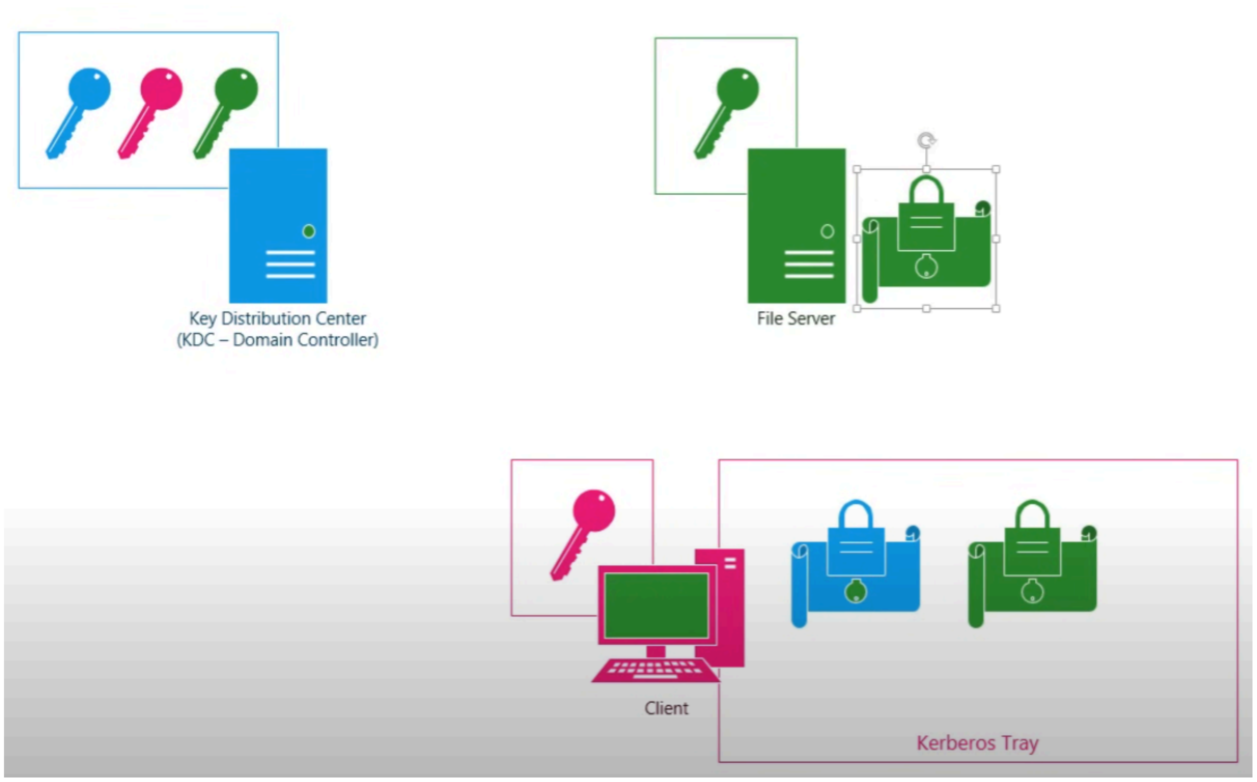


Creates a TGT and uses filer server's password to encrypt it



Sends to client and client stores it into the kerberos tray

Now, for the next 8 hours, anytime the client wants something from the file server, all it has to do is make a copy of this TGT(Green) and send it to the file server.



Client sends TGT to the file server

After receiving TGT from the client, the file server does not know who it is. What it does is, it decrypts the TGT using its own Password and if it decrypts successfully then it must have been generated by the KDC because that's the only other domain member who knows the file server's password. The file server then accepts the TGT that contains information about the client like username, groups that the client belongs from, etc and uses this information to decide what the client is allowed to access. Everytime the client wants to get something from the file server, the client needs to resend a copy of the TGT.
The file server doesn't store the client's TGT(garbage file) in memory because in practical life a file server manages files
for probably 100 and 1000 of users so if it stores the TGT of each and every user then it will be a huge pain.

That's how Kerberos authentication protocol works.

Kerberos protocol is an example of **private key/secret key/symmetric** cryptography.
What is the difference between public key/asymmetric cryptography and private key/secret key/symmetric cryptography?

Ans: **For public key/asymmetric cryptography we use a pair of keys: public key and private key (Example: RSA algorithm discussed earlier).**

For private key/secret key/symmetric cryptography, we use one shared secret key (Example: Kerberos protocol).

For some protocols, we use both symmetric and asymmetric cryptography (Example: SSL protocol for HTTPS).

What is the disadvantage of Kerberos protocol?

Ans: **Single point of failure.**