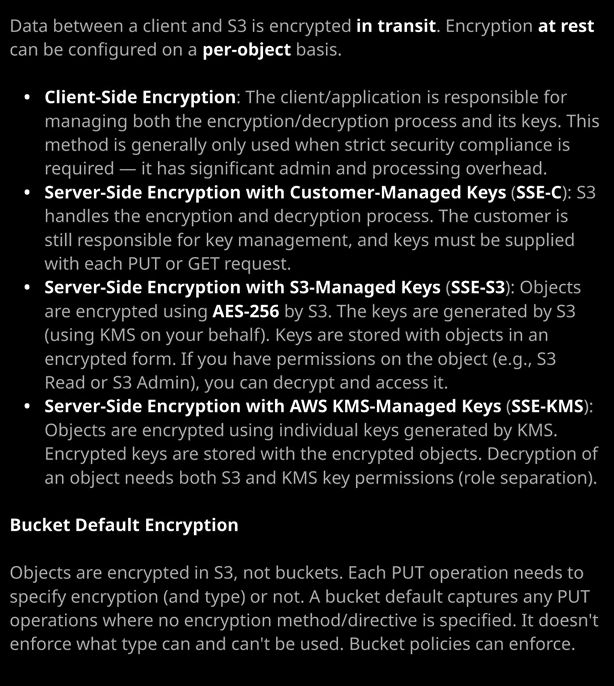
**Encryption within S3**



Now S3, as you know by now is an object storage service. It lets you upload, also know as put objects, they're stored in the bucket and you can access or get these objects at a later stage. **Now data can be encrypted between you and S3 using SSL, and this is known as encryption in transit but it can also be encrypted at rest. By default, objects stored within S3 aren't encrypted at rest.** So whatever you send in the port operation is stored persistently, as is on S3 storage. **If you send plain text then plain text is stored if you send cipher text than encrypted data is stored persistently an S3 by default. Whatever you send, to S3 is stored in history.** But you also have a number of options for encryption at rest when using S3 and it's important for the exam on dhe for real world production usage to understand the architecture of these different options.

So the first tape of encryption is to use what's known as **client side encryption.** Now client side encryption is where you using a client or an application or manually a responsible for managing both the encryption and decryption process as well as its keys. **Now this method is generally only used when strict security compliance is required. Maybe you're using really sensitive data, and you've got a local server based backup software that uses S3 to store it's backup data. And if that's the case, this backup application running on a server might manage its own encryption keys. It encrypts the data before it sends it to S3 until that data is stored directly on S3 in the form that you provide it, which is encrypted.** **So with clients, I'll encryption. It's not really involving S3 in the process, but all your encrypting on decrypting the data.** You're managing the keys and then you're uploading or downloading. So putting or getting that data from S3,

the first type of encryption that utilizes S3 in any way, is known as **server side encryption with customer managed keys, also known as SSE. So service side encryption -c and using SSE-C**. **S3 handles the encryption and decryption process now This is a fairly CPU intensive process. So by offloading this to S3, it means you don't have to run a super powerful machine on your customer premises to encrypt or decrypt the data but with SSE-C you're managing the encryption keys.** Now, what this means in practice is that when you're uploading or putting an object, you give it to S3 You provide the encryption key, S3 performs that encryption and then stores that encrypted data or cipher text on S3 storage and then it discards that key and it means when you want to decrypt that data, you need to make requests to S3 provide the same encryption key back to S3 again. It decrypts that data and sends that data back to you in its original form**. So with SSE-C S3 manage with CPU intensive encryption and decryption process, but you're responsible for managing your keys**. Now, this might be something that you're actively looking to do. So you might want to manage your own encryption keys, maybe again using some really sensitive data. And if you do that, you might need to control your own encryption keys. **The problem, though, is that there is a significant management overhead for keeping track of those keys you need to generate them. You need to manage which ones they used for which files you need to securely store them, back them up and manage rotation. It's a fairly admin heavy process, but you might need to do that if your own security processes require it.**

If that's not the case, if you give over additional responsibility to s three, then you can also **use SSC-S3 or service side encryption with S3 managed keys**. If you was this option, then you're providing to S3 data in its original form. **S3 is managing the encryption and decryption process, and it's done using keys. That S3 managers on your behalf. Essentially, what happens is S3 uses a service called KMS, which is the key management service. It uses one of the keys that the key management service has and managers inside that product. It generates what's known as a data encryption Key. KMS provides two versions of that key, one that's encrypted on one that's decrypted and S3 uses the decrypted version of that key to encrypt the object stores it on S3. It takes the encrypted version off that key, stores it with that object and then discards everything else.** And it means that you always know which key is used to encrypt which object and it's all handled within S3. It's completely transparent to you. It sounds fairly complicated, but it's not.

Let's just step through exactly how that works. I'm going to go into my ac-catpics1337 bucket and I'm going to click on "Upload." I'm going to click on add files on I'm going to select the larry.jpg image that I demonstrated in previous lesson. I'm going to click on open. Then I'll click of next next again and scroll down to our asset. The encryption on an object I'm going to pick to use Amazon S3 master key on when I select this option. That's using SSE-S3. It's essentially working with KMS to manage the encryption process, but it's done. Where S3 managers that end to end? So it's slightly less secure than using the final type of encryption that I'm going to talk about next but it does mean it's less adamant overhead. So click on "Next" how upload that object I'm supplying S3 at this point with that object in a plain text form. So in its original form S3 takes that object, it gets a key from KMS. It encrypts that object with that key and then stores that object along with the encrypted version off that key in persistent storage. Now for select that object and then go to its properties, you'll see for encryption it's using AS256 Now that's the algorithm that you used by S3 to encrypt an object. So if you ever see **AS256** mentioned anywhere, then you know that it's using SSE-S3 and that's important to understand for the exam. That's the algorithm that's used by SEC- necessary.

Now, the last type of encryption that I want to talk about is service side encryption with **AWS KMS managed keys** on the distinction between this type of encryption on the previous type is actually fairly minor, but the benefit that it has is massive. If you's SSE-S3 S3 handles the encryption end to end now. This means that there's a low amount of admin overheads, so it's great to use. It's a really effective way to secure your data without having to manage that process End to end. The problem, though, is that if you give an IAM user the ability to manage, S3 said, essentially S3 admin rights than as well as banal to manage objects in a necessary bucket that individual can also encrypt and decrypt data, and sometimes you want to do what's known as role separation. You want to allow certain identities to be given S3 administrator rights, but not allow them to interact with objects. If you're in a large company, if you deal with sensitive data, then roll separation is something that your security team will be really comfortable with. The way that S3 implements this role. Separation is by using SSE-KMS.

Now let's demonstrate the difference how this works. If I go to upload an object and go to add files and then I'm going to select group.jpg, click "Open" next again next again, scroll down and instead of slighting AWS S3 master key this time. How select AWS Kms Master Key. Now want to do that? I'll be asked to select a customer master via KMS Master Key so straight away I get more functionality because I'm able to select a specific customer master key to use. So what is a custom? A master key? I'll be talking about that more later in the course where I talk about KMS but just to make it easier for this lesson, I'll open up the IAM console and I'm going to go down to encryption keys. This will open up the KMS console, and if I go to AWS managed keys, I'll see a list of keys that use for encryption for various different products. Now what I could do is I can create a customer managed key, so I'll do that. I'll create a key. I'll call the key supersecret cat pics. So this is a key that I'm going to use to encrypt all of my most sensitive cat pictures. So I'll click on "Next," "Next" again, "Next" again, "Next" again, and then I'll finish creating the key. Now we're going to talk about this in more detail later in the course when I talk about KMS but essentially a custom master key is its own object. You're able to create it outside the product which uses it so outside of S3 and I could define permissions specifically on this customer master key. What that means relative to the associate level solutions architect exam is that if then I want to upload objects to a necessary evil care. So oops.jpg I'm able to select that specific supersecret cat pics key to use. Now, in order to do this, **it means that not only do I need the permissions to upload an object to us three, but I also need the permissions to use this master key to encrypt that object. So it's two separate pieces of permissions. The benefit of this is that in order to decrypt the object on our only permissions toe access it but I also need permissions to decrypt it using this key. Now that means we can separate the roles. It means we cannot allow individual identities to have admin rights over S3, but not give them permissions to work on this key and that means we can separate and have S3 administrators and have certain identities or groups that can be encrypting and decrypting data but not have access to S3. It allows this role separation, and that's especially important when you've got really sensitive data. So if I upload this object once it's not loaded, I'm just going to go ahead and select it because you're able to tell that it uses KMS encryption by selecting the object and then under encryption type.** It says AWS-KMS. So remember, SSE-S3 will always say AS256 and if using KMS using your own manage keys, then it'll say AWS-KMS. I just want you to be aware that the technology exists if any questions mention roll separation or mentioned different teams where one team needs to manage. S3 on the other needs to manage encryption. You know, the only type of encryption you can use S3-KMS. Important for the exam is the object.

**Encryption is done on a per object basis. You don't specify bucket encryption.** That's not how it works. It's a per object thing. Every time you upload an object or put an object into a story, you need to specify a type of encryption. If you don't, it won't use any form of encryption. What you are able to do, though, is to specify a default encryption type on a bucket. So I could specify for this bucket here that I want AS256 encryption to be used. If any time an object is uploaded, no specific encryption is specified. Now the important thing to realize about the **default encryption, it does not force a specific type of encryption.** So even though I've got this default encryption set, I could put an object using AWS KMS and I would be allowed to do so if I want to insist on a specific type of encryption than a need to use a bucket policy a bucket policy can allow uploads to occur only if a specific type of encryption is used and I'll make sure that include a link in the lesson description with an example bucket policy showing how this is done. Now, don't be concerned about the level of technical depth in this lesson for the exam.

The only thing that you need to understand is that **client side encryption is where you encrypt and decrypt data before it goes to or from S3 and that you're managing both the encryption. The decryption on the keys, service side encryption with customer managed keys. This is SSE-C and this is where you provide the object to S3 as well as the key S3 encrypts Using that key, it discards the key and to decrypt it, you need to supply the key that she used for that decryption process. Also for the exam SSE-C does have some limitations. *So a lot of other features available in S3 such as cross region replication that I'll be talking about later in the course does not support the use of SSE-C*, generally, if he using encryption with S3, you're either using SSE-S3 or SSE-KMS of those. The two options on the one that your pick will depend on whether you need this role Separation. If you need the role separation, we need to be able to manually rotate your own keys. You need to use SSE-KMS. If you see a question that requires AES256 you don't see any other information than use SSE-S3 with that being said, that's all I wanted to cover in this lesson.**