100 XP

Guidelines for using Azure Key Vault

5 minutes

Azure Key Vault is a centralized cloud service for storing application secrets such as encryption keys, certificates, and server-side tokens. Key Vault helps you control your applications' secrets by keeping them in a single central location and providing secure access, permissions control, and access logging.

There are three primary concepts used in an Azure Key Vault: *vaults*, *keys*, and *secrets*.

Vaults

You use Azure Key Vault to create multiple secure containers, called vaults. Vaults help reduce the chances of accidental loss of security information by centralizing application secrets storage. Organizations will have several key vaults. Each key vault is a collection of cryptographic keys and cryptographically protected data (call them "secrets") managed by one or more responsible individuals within your organization. These key vaults represent the logical groups of keys and secrets for your organization; those that you want to manage together. They are like folders in the file system. Key vaults also control and log the access to anything stored in them.

You can create and manage vaults using command line tools such as Azure PowerShell or the Azure CLI, using the REST API, or through the Azure portal.

For example, here's a sample Azure CLI command line to create a new vault in a resource group:

Azure CLI | Copy

```
az keyvault create \
    --resource-group <resource-group> \
    --name <your-unique-vault-name>
```

Here's the same command using Azure PowerShell:

PowerShell | Copy

```
New-AzKeyVault -Name <your-unique-vault-name> -ResourceGroupName <resource-group>
```

Keys

Keys are the central actor in the Azure Key Vault service. A given key in a key vault is a cryptographic asset destined for a particular use such as the asymmetric master key of Microsoft Azure RMS, or the asymmetric keys used for SQL Server TDE (Transparent Data Encryption), CLE (Column Level Encryption) and Encrypted backup.

Microsoft and your apps don't have access to the stored keys directly once a key is created or added to a key vault. Applications must use your keys by calling cryptography methods on the Key Vault service. The Key Vault service performs the requested operation within its hardened boundary. The application never has direct access to the keys.

Keys can be single instanced (only one key exists) or be versioned. In the versioned case, a key is an object with a primary (active) key and a collection of one or more secondary (archived) keys created when keys are rolled (renewed). Key Vault supports asymmetric keys (RSA 2048). Your applications may use these for encryption or digital signatures.

There are two variations on keys in Key Vault: hardware-protected, and software-protected.

Hardware protected keys

The Key Vault service supports using HSMs that provide a hardened, tamper-resistant environment for cryptographic processing and key generation. Azure has dedicated HSMs validated to FIPS 140-2 Level 2 that Key Vault uses to generate or store keys. These HSM-backed keys are always locked to the boundary of the HSM. When you ask the Key Vault service to decrypt or sign with a key, the operation is performed inside an HSM.

You can import keys from your own hardware security modules (HSMs) and transfer them to Key Vault without leaving the HSM boundary. This scenario is often referred to as *bring your own key*, or BYOK. More details on generating your own HSM-protected key and then transferring it to Azure Key Vault is available in the summary of this module. You can also use these Azure HSMs directly through the Microsoft Azure Dedicated Hardware Security Module (HSM) service if you need to migrate HSM-protected apps or maintain a high security compliance requirement.

Software protected keys

Key Vault can also generate and protect keys using software-based RSA and ECC algorithms. In general, software-protected keys offer most of the features as HSM-protected keys except the FIPS 140-2 Level 2 assurance:

- Your key is still isolated from the application (and Microsoft) in a container that you manage

- It's stored *at rest* encrypted with HSMs
- You can monitor usage using Key Vault logs

The primary difference (besides price) with a software-protected key is when cryptographic operations are performed, they are done in software using Azure compute services while for HSM-protected keys the cryptographic operations are performed within the HSM.

> **Tip**
>
> For production use, it's recommended to use HSM-protected keys and use software-protected keys in only test/pilot scenarios. There is an additional charge for HSM-backed keys per-month if the key is used in that month. The summary page has a link to the pricing details for Azure Key Vault.

You determine the key generation type when you create the key. For example, the Azure PowerShell command `Add-AzureKeyVaultKey` has a `Destination` parameter that can be set to either `Software` or `HSM`:

PowerShell Copy

```
$key = Add-AzureKeyVaultKey -VaultName 'contoso' -Name 'MyFirstKey' -Destination
'HSM'
```

Secrets

Secrets are small (less than 10K) data blobs protected by a HSM-generated key created with the Key Vault. Secrets exist to simplify the process of persisting sensitive settings that almost every application has: storage account keys, .PFX files, SQL connection strings, data encryption keys, etc.

Key vault uses

With these three elements, an Azure Key Vault helps address the following issues:

- **Secrets management**. Azure Key Vault can securely store (with HSMs) and tightly control access to tokens, passwords, certificates, API keys, and other secrets.
- **Key management**. Azure Key Vault is a cloud-based key management solution, making it easier to create and control the encryption keys used to encrypt your data. Azure services such as App Service integrate directly with Azure Key Vault and can decrypt secrets without knowledge of the encryption keys.

- **Certificate management**. Azure Key Vault is also a service that lets you easily provision, manage, and deploy public and private SSL/TLS certificates for use with Azure and your internal connected resources. It can also request and renew TLS certificates through partnerships with certificate authorities, providing a robust solution for certificate lifecycle management.

> **ⓘ Important**
>
> **Key Vault is designed to store configuration secrets for server applications.** It's not intended for storing data belonging to your app's users, and it shouldn't be used in the client-side part of an app. This is reflected in its performance characteristics, API, and cost model.
>
> User data should be stored elsewhere, such as in an Azure SQL database with Transparent Data Encryption, or a storage account with Storage Service Encryption. Secrets used by your application to access those data stores can be kept in Key Vault.

Best practices

Here are some security best practices for using Azure Key Vault.

Best practice	Solution
Grant access to users, groups, and applications at a specific scope.	Use RBAC's predefined roles. For example, to grant access to a user to manage key vaults, you would assign the predefined role Key Vault Contributor to this user at a specific scope. The scope, in this case, would be a subscription, a resource group, or just a specific key vault. If the predefined roles don't fit your needs, you can define your own roles.
Control what users have access to.	Access to a key vault is controlled through two separate interfaces: management plane, and data plane. The management plane and data plane access controls work independently. Use RBAC to control what users have access to. For example, if you want to grant an application the rights to use keys in a key vault, you only need to grant data plane access permissions by using key vault access policies, and no management plane access is needed for this application. Conversely, if you want a user to be able to read vault properties and tags but not have any access to keys, secrets, or certificates, you can grant this user read access by using RBAC, and no access to the data plane is required.

Best practice	Solution
Store certificates in your key vault.	Azure Resource Manager can securely deploy certificates stored in Azure Key Vault to Azure VMs when the VMs are deployed. By setting appropriate access policies for the key vault, you also control who gets access to your certificate. Another benefit is that you manage all your certificates in one place in Azure Key Vault.
Ensure that you can recover a deletion of key vaults or key vault objects.	Deletion of key vaults or key vault objects can be either inadvertent or malicious. Enable the soft delete and purge protection features of Key Vault, particularly for keys that are used to encrypt data at rest. Deletion of these keys is equivalent to data loss, so you can recover deleted vaults and vault objects if needed. Practice Key Vault recovery operations regularly.

Note

If a user has contributor permissions (RBAC) to a key vault management plane, they can grant themselves access to the data plane by setting a key vault access policy. It's recommended that you tightly control who has contributor access to your key vaults, to ensure that only authorized persons can access and manage your key vaults, keys, secrets, and certificates.

Next unit: Manage access to secrets, certificates, and keys

Continue >