Salesforce Shield Platform Encryption Architecture

Version 60.0, Spring ’24

Last updated: March 22, 2024

CONTENTS

CHAPTER 1 Why Encrypt

In this chapter ...

Security and trust are major factors in every company's evaluation of public cloud services. Salesforce customers choose which business functions to run on the Salesforce Platform, which applications they can build to extend those functions, and what data they must store to enable those functions.

The Verizon 2019 Data Breach Investigations Report counted over 41,000 security incidents and over 2,000 confirmed data breaches worldwide in 2018. A study of 944 data breach incidents in the first half of 2018 showed that encryption was used to secure breached data in only 2.2% of cases. According to some estimates, over 14 billion records have been compromised since 2013.

While Transport Layer Security (TLS) is one effective tool against data loss, researchers in the security community have demonstrated numerous techniques to compromise TLS and other encryption solutions. Across the industry, there's an increasing awareness that transport security isn't enough to protect sensitive data. Additionally, attackers are targeting a wider array of targets. In the past, attacks focused on high-value financial targets — retail and point of sale, credit card processors, card issuers, and banks. Now, attackers are stealing any personally identifiable information (PII), which can enable further social engineering attacks and more significant compromises.

Customers increasingly use the Salesforce Platform to build applications that require PII and other sensitive, confidential, or proprietary data. Standard features such as authentication and single sign-on, granular access controls, and activity monitoring give most customers enough control over when and how they protect their data. But when sensitive data is stored on the Salesforce Platform, some customers want additional layers of protection on top of our standard security measures.

Balance Data Security with Business Needs

Choosing to store PII, sensitive, confidential, or proprietary data with any third party often prompts customers to more closely investigate both external regulatory and internal data compliance policies. Internal policies frequently rely on interpretation of external regulations.

As customers look at regulations such as PCIDSS, HIPAA/HITECH, and FedRAMP through the lens of cloud-based service adoption, they typically take a pragmatic but conservative approach to data protection in the cloud

This pragmatic approach includes three requirements shared by a wide variety of customers in regulated industries such as financial services, healthcare, and life sciences, as well as manufacturing, technology, and government.

1. Encrypt sensitive data when it's stored at rest in the Salesforce cloud.
2. Support customer-controlled encryption key lifecycles.
3. Preserve application and Salesforce Platform functionality.

However, there's a tradeoff between strong security and functionality. Data encrypted at rest can make preserving Salesforce functionality difficult, if not impossible. The degree depends on where encryption and decryption occur and where the encryption keys are stored. What the business wants often differs from what security and compliance require.

Salesforce Encryption Principles

To balance security demands with customers' functional requirements, Salesforce defined a set of principles that drove our decisions around solution design and architecture. We focused on the problems we wanted to solve, clearly defined the boundaries of our solution, and identified the implications and tradeoffs of the design.

Encrypt data at rest

The Salesforce Shield Platform Encryption solution encrypts data at rest when stored on our servers, in the database, in search index files, and the file system. To encrypt data at rest and preserve functionality, we built the encryption services natively into the Salesforce Platform.

Natively integrate encryption at rest with key Salesforce features

One of the things that makes the Lightning Platform so remarkable is that it's driven by metadata. Shield Platform Encryption uses that metadata to tell the other platform features which data is encrypted. This way we can prevent those features from inadvertently exposing plaintext or ciphertext. And we can ensure that critical business functionality — like partial search — continues to work even when data is encrypted.

Use strong, flexible encryption

The Shield Platform Encryption solution uses strong, probabilistic encryption by default on data stored at rest. Shield Platform Encryption uses the Advanced Encryption Standard (AES) with 256-bit keys using CBC mode and a random initialization vector (IV). While this type of encryption results in a loss of some functionality, such as sort operation, we consider such losses a reasonable tradeoff in favor of security.

However, we recognize that in some cases, business requirements depend on preserving more functionality, which can influence what data customers decide to encrypt. In those cases, we offer deterministic encryption. Deterministic encryption uses the Advanced Encryption Standard (AES) with 256-bit keys using CBC mode and a customer- and field-specific static IV. This static IV is a hash of the entity ID, field ID, and key ID, making it unique to each customer and field per org. Upon key rotation, a new static IV is generated for each field. When data is encrypted deterministically, the application server computes the unique static IV for each field, and then uses the static IV to generate ciphertext. Because the ciphertext for each field includes bits that associate it with a specific field, customer queries return encrypted values in that field. In this way, deterministic encryption only decreases encryption strength as much as is necessary to allow filtering.

Let customers drive the key lifecycle

We built a key management framework that scales to our multitenant model and gives you complete control over the key management lifecycle. Since the encryption service is built natively into the Salesforce Platform, the encryption keys can reside in the Salesforce environment. Or, when accessible by the Salesforce Shield Key Management Service (KMS), in an external key store. Adhering to the principle that customers must have complete control over the key lifecycle, we built key management functionality into the Setup UI and API. Customers decide when to generate, supply, rotate, import, export, and destroy keys. Customers also determine who is responsible for performing these tasks. With the Bring Your Own Keys (BYOK) option, you can generate and store key material outside of Salesforce using your own crypto libraries, enterprise key management systems, or hardware security modules. As with all administration tasks, everything is audited.

Protect keys from unauthorized access

A primary consideration when architecting our key management infrastructure was making encryption keys available to the encryption service while preventing privileged Salesforce employees, such as DBAs, from inappropriately accessing them. This consideration led us to incorporate hardware security modules (HSMs) into the infrastructure. Shield Platform Encryption uses HSMs to generate cryptographic key material. The result is a shared key management service that creates tenant-specific encryption keys. These keys aren't persisted; they're therefore inaccessible to Salesforce employees and, by extension, malicious external attackers.

Encrypt as little data as possible

Our design gives customers control over what data they encrypt. Your administrator chooses whether to turn on encryption for standard fields, custom fields, files, attachments, and more. You also choose which specific fields to encrypt at rest. The driving principle is to encrypt as little as possible to preserve functionality while keeping private, sensitive, confidential, and regulated data safe.

Before You Encrypt

Before you decide to encrypt data in Salesforce — or in any cloud service — first make sure you're matching the right security solution to the type of threats you face.

For example, if you're concerned about protecting against end-user or administrative account takeover attacks, data encryption might not be an appropriate control against such a threat. Takeover attacks usually occur through social engineering and malware infection. Consider instead malware detection and activity monitoring as ways to identify when users could have been compromised and a malicious outsider is attempting to gain access to data.

Salesforce Shield Platform Encryption protects data at rest. It shouldn't be confused with a control that encrypts data in transit, such as Transport Layer Security (which Salesforce provides by default for your org).

Shield Platform Encryption is best suited for:

- Protecting against data loss due to unauthorized database access
- Bolstering compliance with regulatory requirements or internal security policies
- Satisfying contractual obligations to handle sensitive and private data on behalf of customers

The best approach is adopting a defense-in-depth strategy that takes advantage of all the security features Salesforce offers. Take the Security Basics Trail to learn about the available customer-controlled security capabilities.

After completing a threat modeling exercise, use the outcome to inform a granular data classification. (Keep the Salesforce Security Guide handy as you work through this project.) Identify data elements that are sensitive, private, or confidential. Your best strategy is to encrypt only the most sensitive of those data elements. Such strategies help balance stronger data protection controls against the need to build and preserve critical business functionality on the Salesforce Platform.

What Gets Encrypted?

Shield Platform Encryption allows you to encrypt a variety of fields, files, and data. We use metadata to keep information in these files and fields secure while preserving the ability to perform common business tasks.

In contrast to Classic Encryption, which uses a custom field type in the Salesforce data model, Shield Platform Encryption makes more fields, files, and data elements available for encryption every year. For the complete list of fields you can encrypt with Shield Platform Encryption, see What Can You Encrypt? in Salesforce Help.

This table compares the features of Shield Platform Encryption and Classic Encryption:

Feature	Classic Encryption	Platform Encryption
Pricing	Included in base user license	Additional fee applies
Encryption at Rest	✓	✓
Native Solution (No Hardware or Software Required)	✓	✓
Encryption Algorithm	128-bit Advanced Encryption Standard (AES)	256-bit Advanced Encryption Standard (AES)
HSM-based Key Derivation		✓
Manage Encryption Keys Permission		✓
Generate, Export, Import, and Destroy Keys	✓	✓
PCI-DSS L1 Compliance	✓	✓
Masking	✓	
Mask Types and Characters	✓	
View Encrypted Data Permission Required to Read Encrypted Field Values	✓	
Encrypted Standard Fields		✓
Encrypted Attachments, Files, and Content		✓
Encrypted Custom Fields	Dedicated custom field type, limited to 175 characters	✓
Encrypt Existing Fields for Supported Custom Field Types		✓
Search (UI, Partial Search, Lookups, Certain SOSL Queries)		✓
API Access	✓	✓
Available in Workflow Rules and Workflow Field Updates		✓
Available in Approval Process Entry Criteria and Approval Step Criteria		✓

CHAPTER 2 How Shield Platform Encryption Works

In this chapter ...

To meet the security requirements of customers while preserving functionality and performance in our multitenant environment, we built the encryption service directly into the Lightning Platform.

The platform's object-relational mapping model includes metadata that describes exactly which data is encrypted. Encrypted data is stored with additional information that uses strong, nondeterministic cryptography supported by the Java Cryptographic Extension (JCE). Encryption and decryption occur in the platform's application layer as application components are materialized by the runtime engine, ensuring that encrypted data doesn't persist in plaintext.

The Shield Key Management Service derives keys on demand from key material generated by HSMs. Customers can opt out of key derivation and supply a final data encryption key. Key material is never persisted. Finally, the architecture supports the simultaneous use of multiple encryption keys, enabling customers to quickly rotate and archive keys without losing access to their data.

Encryption Happens in the Application Layer

The Lightning Platform's foundation is a metadata-driven software architecture that enables multitenant applications.

Application components, such as Salesforce objects, aren't modeled directly in our underlying relational database. Instead, when customers interact with their data in a Salesforce application, the platform's runtime engine materializes the data using metadata stored separately in the Lightning Platform's Universal Data Dictionary (UDD).

This way, each tenant's data is kept secure in the shared database, tenants can customize schema in real time without affecting other tenants' data. Also, the application's code base can be patched or upgraded without breaking tenant-specific customizations. See Platform Multitenant Architecture for details.

The UDD includes metadata that determines which data is encrypted at runtime. The encryption service works in the Lightning Platform's application layer. That is, data is encrypted directly before it's stored in the database. The resulting encrypted payload is stored with metadata about the specific key used to encrypt it. For decryption, data is decrypted as it's materialized. It's then pushed up through the application pipeline and appears in plaintext to the user who requested it.

By embedding the encryption metadata in the UDD, the Shield Platform Encryption architecture allows customers to choose what data to encrypt. The Shield Platform Encryption engine strictly manages the flow of encrypted data from the application layer to the database and vice versa. The relevant code paths run through the UDD before data is read or stored.

Cryptographic Library and Algorithms

Shield Platform Encryption uses the Java Cryptography Extension (JCE) to encrypt and decrypt data. Specifically, Shield Platform Encryption uses the Advanced Encryption Standard (AES- 256) in CBC mode with, by default, a random IV. Deterministic encryption uses a field-specific static IV, which is a hash of the entity ID, field ID, and key ID. This method makes the static IV unique to each customer and field per org. Upon key rotation, a new static IV is generated for each field.

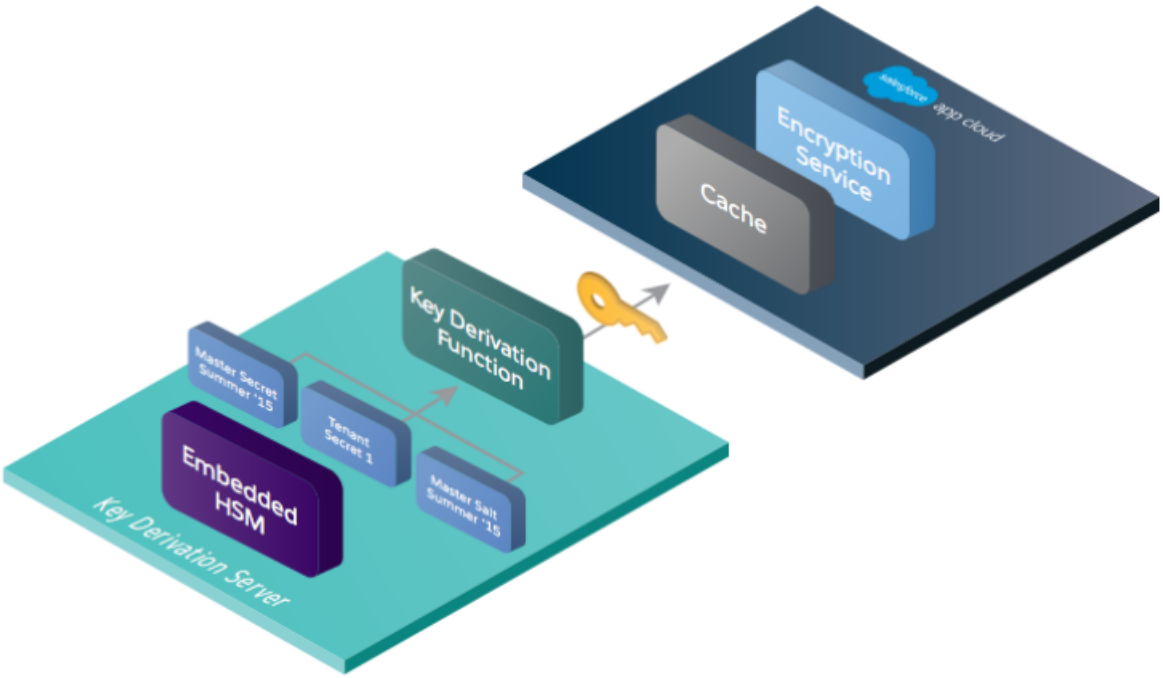
Encrypted Key Cache

Salesforce has taken steps to reduce access to encryption key material across the servers where encrypted keys are available for the application servers. Rather than holding data encryption keys on the application server, they're temporarily held in an encrypted key cache. This feature is a central service that all application servers can access. When the customer accesses encrypted data on the application server, the server uses a data encryption key for a short time to perform both encryption and decryption operations. After this transaction is complete, related memory containing the data encryption key is freed. Thus, the data encryption keys aren't cached on the application server.

Data Encryption Keys

The AES-256 keys used to encrypt customer data aren't persisted. Instead, they're either derived on demand or, if customers opt out of derivation, supplied by the customer. Customer-supplied keys can be stored encrypted and protected by the HSM tier in the database. Or, they can be fetched on-demand from a customer-specified external key management service.

Derived keys are generated on demand from secrets generated by logically and physically separated HSMs. The master secret is generated at the start of each Salesforce release and stored securely in Salesforce's internal file system.



The customer-specific key material is supplied by customers or generated by customers on demand, and then stored securely in the database. These secrets, along with a master salt generated at the start of each release, are used as inputs to Password-Based Key Derivation Function 2 (PBKDF2) to derive data encryption keys. PBKDF2 is run on the Shield Key Management Service (KMS) in a Salesforce data center. Data encryption keys are sent in (encrypted) form back to the encryption service that runs on the Salesforce Platform and stored are in the encrypted key cache.

The organization's specific search index key is different than the data encryption key. See the Search Encryption at Rest Process Flow section for more information.

Storing Encrypted Payloads

Encrypted data is stored in the database with its metadata.

The metadata includes:

- A bit that indicates the field contains ciphertext
- The ID of the customer's key material used to derive the matching encryption key (if the customer hasn't opted out of key derivation)
- A random or static 128-bit initialization vector (IV)

The key material's ID is used to locate the key value and creation date. These values are stored in a Salesforce object called TenantSecret. When a user accesses or saves encrypted data, the encryption service sends a request to the Shield KMS. KMS then uses the customer's key material and corresponding master secret, identified by the key material's generation or upload date, to derive a data encryption key. The random IV is used with the encryption key to non-deterministically encrypt the data.

Shield Platform Encryption Process Flow

Before data is encrypted, a Salesforce administrator must enable encryption and generate or supply key material. For each field, file, attachment, and data element on which encryption is enabled, the corresponding metadata in the UDD is updated to reflect the new encryption setting.

1. When a user saves encrypted data, the runtime engine determines from metadata whether the field, file, attachment, or data element must be encrypted before storing it in the database.
2. If the decision is to encrypt, the encryption service checks for the matching data encryption key in the encrypted key cache.
3. The encryption service determines if the key exists.
   a. If yes, the encryption service retrieves the key.
   b. If no, the service sends a derivation request to the Shield KMS and returns it to the encryption service running on the Lightning Platform. Data moving between the Shield KMS and the encryption service is encrypted by the TLS protocol, which uses a certificate signed by a dedicated Salesforce authority. This certificate's private key is stored locally in an encrypted form. The certificate's public and private keys are rotated regularly.
4. After retrieving or deriving the key, the encryption service generates a random initialization vector (IV) and encrypts the data using JCE's AES-256 implementation.
5. The ciphertext is saved in the database or file storage. The IV and corresponding ID of the key material used to derive the data encryption key are saved in the database.

Search Encryption at Rest Process Flow

The Salesforce search engine is built on the open-source enterprise search platform software Apache Solr. The search index, which stores tokens of record data with links back to the original records stored in the database, is housed within Solr. Partitions divide the search index into segments to allow Salesforce to scale operations. Apache Lucene is used for its core library.

Using Shield Platform Encryption's HSM-based key derivation architecture, metadata, and configurations, Search Encryption at Rest runs automatically when Shield Platform Encryption is in use. The solution applies strong encryption on an org-specific search index .fdt, .tim, and .tip file types using an org-specific AES-256 bit encryption key. The search index is encrypted at the search index segment level, and all search index operations require index blocks to be encrypted in memory.

The only way to access the search index or the encrypted key cache is through programmatic APIs.

Before the search index files are encrypted, a Salesforce security administrator must enable Search Encryption at Rest. The administrator then generates or uploads their key material specifically for search index files, and sets up their encryption policy. This policy determines which data elements must be embedded with encryption. The admin configures Shield Platform Encryption by selecting fields and files to encrypt. An org-specific key specifically for search index encryption is derived from the tenant secret or customer-supplied key material on demand if the customer doesn't opt out of derivation. The key material is passed to the search engine's cache on a secure channel.

When a User Creates or Edits Records

1. The core application determines if the search index segment must be encrypted or not based on metadata.
2. If the search index segment is to be encrypted, the encryption service checks for the matching search encryption key ID in the encrypted key cache.
3. The encryption service determines if the key exists in the encrypted key cache.
4. If the key exists in the encrypted key cache, the encryption service uses the key for encryption.
5. If not, the service sends a request to the core application, which in turn sends an authenticated derivation request to the Shield KMS and returns the key to the core application server. Data moving between the Shield KMS and the encryption service is encrypted by the TLS protocol, which uses a certificate signed by a dedicated Salesforce authority. This certificate's private key is stored locally in an encrypted form. The certificate's public and private keys are rotated regularly.
6. After retrieving the key, the encryption service generates a random initialization vector (IV) and encrypts the data using JCE's AES-256 implementation.
7. The key ID (identifier of the key being used to encrypt the index segment) and IV are saved in the search index.

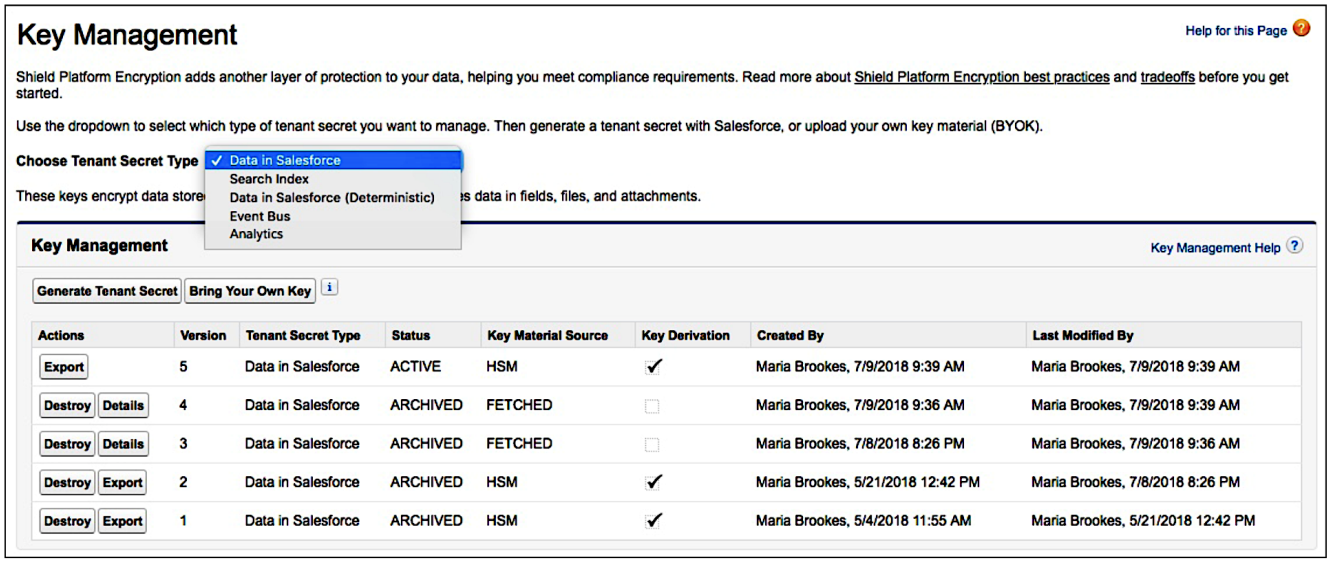
When a User Searches for an Encrypted Term

1. The term is passed to the search index, along with which Salesforce objects to search.
2. When the search index executes the search, the encryption service opens the relevant segment of the search index in memory and reads the key ID and IV.
3. Repeats steps 3 through 5 in the previous search index encryption process.
4. The search index processes the search and returns the results to the user seamlessly.

If Salesforce administrators disable encryption on a field, all index segments that were encrypted are unencrypted and key ID is set to null. This process can take up to seven days.

Key Management

Shield Platform Encryption allows Salesforce administrators to manage the lifecycle of their data encryption keys while protecting those keys from unauthorized access. To ensure this level of protection, data encryption keys are never persisted on disk.



The master secret is generated by a master hardware security module (HSM) at the start of each release. The master HSM is "air-gapped" from Salesforce's production network and stored securely in a bank safety deposit box. Only designated Salesforce security officers can access the safety deposit box and the master HSM stored within.

Key material is either generated on demand using HSMs embedded in the Shield KMS, or supplied by the customer using the Bring Your Own Key (BYOK) service.

The Bring Your Own Key service, introduced in Winter '17, gives customers more control and flexibility for managing key material via an API service. Customers can use open-source crypto libraries, their existing HSM infrastructure, or even third-party key brokering services to create and manage tenant secrets and data encryption keys outside of Salesforce. They can then give Salesforce's KMS access to that key material. Customers can revoke this access at any time.

The Shield KMS has access to the release-specific secrets for every Salesforce release. By default, when a data encryption key is needed to encrypt or decrypt customer data, the Shield KMS derives the key from the master and tenant secrets. Customers can opt out of key derivation on a key-by-key basis and upload a final data encryption key, or store their keys in external key management systems for on-demand retrieval. By controlling the lifecycle of your organization's key material, you control the lifecycle of the derived data encryption keys. Your Salesforce administrator specifies a user to manage the key material for your organization and assigns that user the Manage

Encryption Keys user permission. This user permission allows the key administrator to generate, supply, archive, export, import, and destroy key material.

It's possible to have more than one active tenant secret or data encryption key in an org. You can apply specific keys to data stored in different areas of Salesforce. For example, search index files are stored separately from other Salesforce data, so customers can apply key material to specific data in those files. Only the most recent tenant secret or data encryption key of a given type is active. Only that key material is used to derive the data encryption key used to encrypt data of a specified type. When you generate or supply key material, the active secret becomes archived. Archived key material is used to decrypt data that was last encrypted when the archived key material was active.

You can destroy an archived tenant secret or data encryption key. If you destroy a tenant secret it's no longer possible to derive the encryption key required to decrypt the data that was encrypted using that key. Similarly, when you destroy a customer-supplied data encryption key, you can't access data encrypted with that key. Take special care to back up and protect both archived key material and encrypted data. After you destroy key material, it's fully removed from the persistent layer and encrypted key cache, and can't be recovered.

Rotating Keys and Re-Encrypting Data

Generating or supplying new key material is called key rotation. When you rotate key material, any resulting derived data encryption keys rotate as well. New data is encrypted and decrypted using the final data encryption key, which is derived by default from the new, active tenant secret. If you opt out of key derivation with BYOK, rotating your key encrypts new data with your active customer-supplied final data encryption key. Existing data stays encrypted with the former key material, even if that key is derived.

Customers can use the self-service background encryption service to traverse the database and file storage, decrypt existing encrypted data, and then re-encrypt the data using the new data encryption key. This process is transparent to users and administrators and must be initiated by Salesforce support.

CHAPTER 3 Data Encryption Key Derivation

In this chapter ...

By default, Shield Platform Encryption uses the Shield Key Management Service to derive data encryption keys for encrypting customer data at rest. Data encryption keys are derived from fragmented secrets that are securely wrapped and stored in Salesforce's internal file system. Key derivation ensures that the keys are never persisted in their composite forms, and enables customers to control the key lifecycle.

Key Derivation Architecture

Secrets and secret-wrapping keys used in the key derivation process are initialized by a master HSM at the start of each release. For customer-driven tenant secrets, they're initialized on demand in production environments by HSMs embedded in the Shield KMS.

BYOK customers can opt out of the key derivation process by uploading their own data encryption keys. Customer-supplied data encryption keys are used to directly encrypt and decrypt data and allows customers more fine-grained control over the key material used to secure their data. Because these keys are applied directly to customer data, Salesforce recommends customers take adequate safety measures to back up and control access to these keys.

Processes in Key Derivation Architecture

HSM initialization

Before they're put to use, both the master and embedded HSMs are initialized, which includes the creation of their respective encryption key pairs.

Per-release secret generation

At the start of each release, the master HSM is plugged into an offline laptop and used to generate the per-release secrets. The secrets are hashed and stored in Salesforce's internal file system for consumption by the embedded HSMs.

Shield KMS startup

When the Shield KMS starts up, it accesses each release's encrypted secrets in the internal file system. Then it decrypts the secrets and stores them in the encrypted key cache in preparation for key derivation.

On-demand tenant secret generation

One of the inputs into the key derivation function that creates your organization's data encryption key is an org-specific, customer-managed tenant secret. Customers control the lifecycle of their data encryption keys by generating new tenant secrets. When a customer generates a new tenant secret, the request is sent to the Shield KMS from the application server and authenticated. Then, an embedded HSM generates a tenant secret, which is encrypted by the Shield KMS and sent back to the application server to be stored in the database.

Key derivation in production environments

When a customer attempts to read or write encrypted data and the corresponding data encryption key isn't cached, the application server sends a derivation request to the Shield KMS. The Shield KMS authenticates the request and derives the key using the secrets in the encrypted key cache. The key is then transmitted securely back to the application server.

HSM Initialization, Secret Generation, and Key Derivation Components

Master HSM (SafeNet® Luna G5, manufactured by Gemalto®)

A FIPS 140-2 hardware-compliant USB device that generates per-release secrets and secret-wrapping keys, and signs the public keys of embedded HSMs. The master HSM is air-gapped from the network at all times and stored in a bank safety deposit box. Access to the master HSM is restricted to designated Salesforce security officers.

Offline Laptop

A machine that the master HSM plugs into while in use. The offline laptop exports the secrets and keys generated by the master HSM to the internal Salesforce file system.

Embedded HSMs (SafeNet® Luna PCI-E, manufactured by Gemalto®)

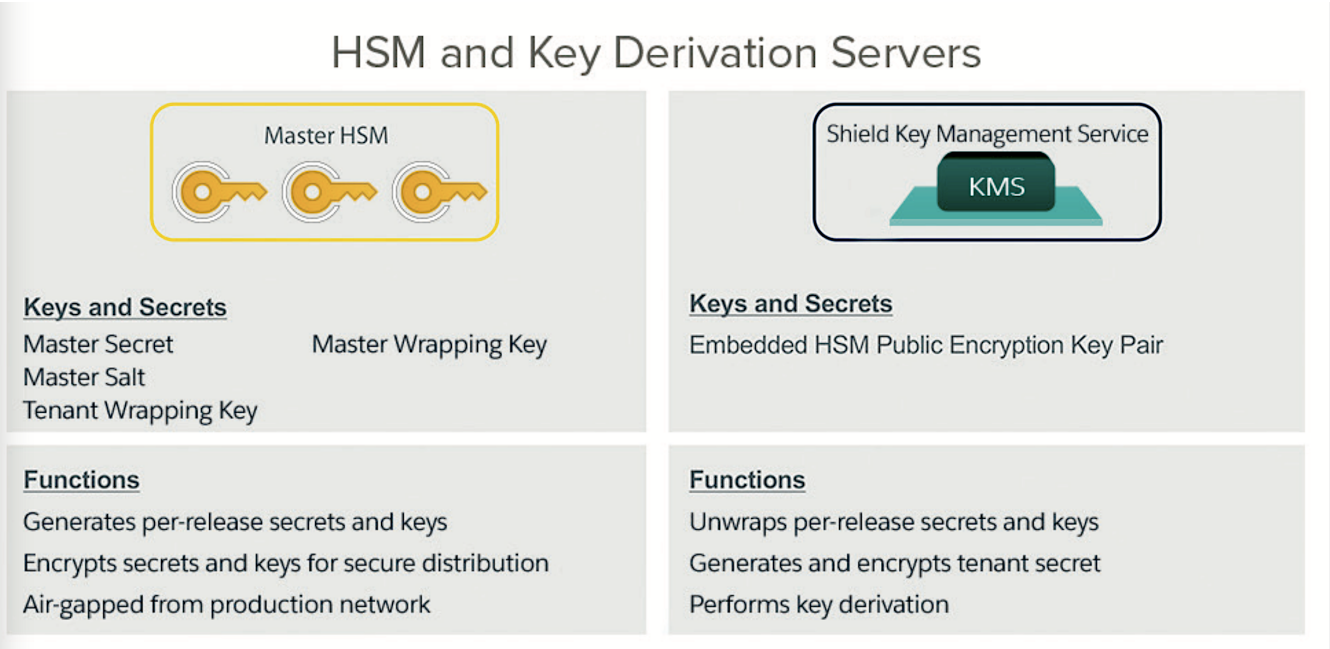
FIPS 140-2 hardware-compliant PCI devices that are plugged into the Shield KMS in Salesforce data centers. Embedded HSMs unwrap secrets that were generated by the master HSM, encrypted, and exported to the Salesforce internal file system. They also generate tenant secrets, the customer-managed fragments of data encryption keys.

Shield KMS
Clusters of load-balanced servers deployed to Salesforce's production data centers that derive data encryption keys from master secrets and tenant secrets.

Application Servers
Servers in production environments that run Salesforce. When a customer attempts to read or write encrypted data or generate a tenant secret, the application server communicates with the Shield KMS to process the request.

Salesforce Internal File System and Source Control
The location and source control mechanism for storing encrypted secrets and their hashes.

Salesforce Search Index
Servers in production environments that manage Salesforce searches. When a user attempts to query encrypted data, the search index processes the request.



HSM Initialization

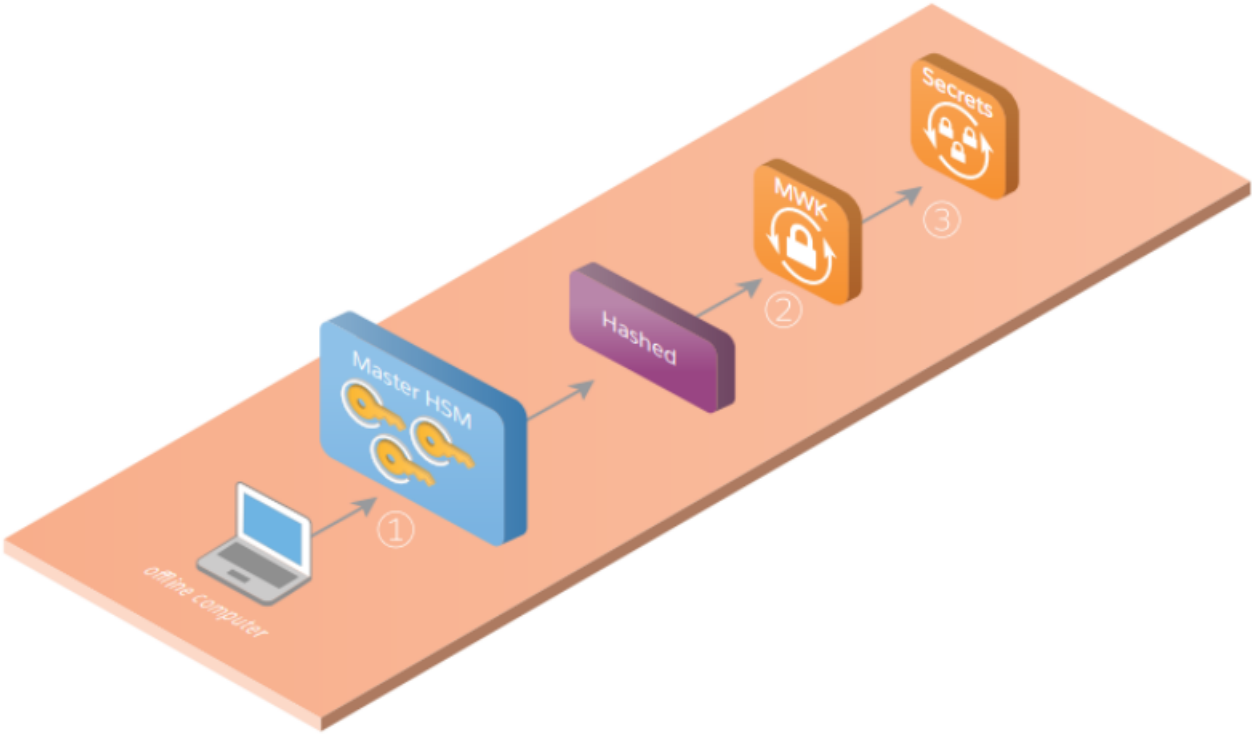
The master HSM and the embedded HSMs must be initialized before they're used. For the master HSM, initialization creates a master HSM encryption key pair and a master HSM signing key pair. For each embedded HSM, initialization creates an embedded HSM encryption key pair.

The master HSM public signing key is used to sign and verify each embedded HSM's public encryption key. At the start of each release, the master and embedded HSM public encryption keys are used to separately encrypt a per-release master wrapping key. This key is in turn used to encrypt the remainder of the per-release secrets used to derive data encryption keys.

This way, each embedded HSM is able to securely access the master wrapping key for each release, which it uses to access the rest of the per-release secrets needed for key derivation. The private keys in each pair are accessible only inside their respective HSMs.

Per-Release Secret Generation

At the start of each release, the master HSM is plugged into the offline laptop and used to generate the per-release secrets and keys (on the HSM itself).



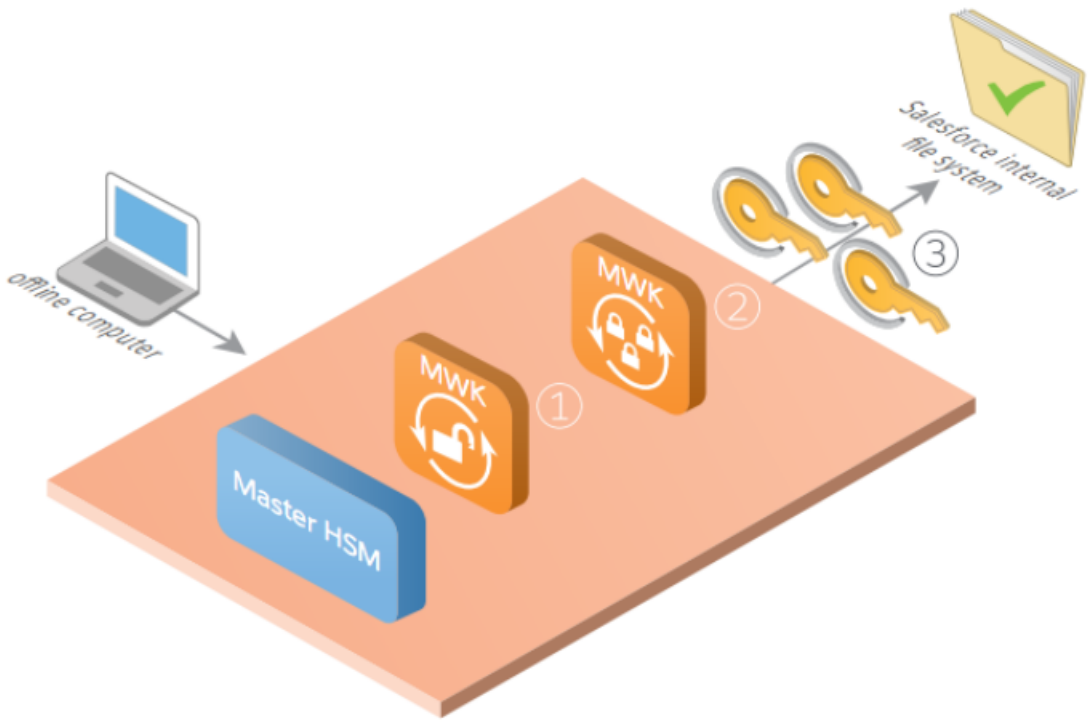
1. The master HSM generates the following secrets:
   - Master secret
   - Master salt
   - Master wrapping key
   - Tenant wrapping key

   Each secret is hashed using SHA-256. For definitions of each secret and key, refer to Keys and Secrets.
2. The master wrapping key (MWK) is encrypted with the master HSM public encryption key and stored locally on the laptop along with its hash.
3. The other secrets are encrypted with the master wrapping key and stored on the laptop with their hashes.

Per-Release Secret Export

After all the secrets are hashed and encrypted, they're checked into source control and exported to the Salesforce internal file system. The plaintext master wrapping key is encrypted with each embedded HSM's public encryption key, checked into source control, and stored in the Salesforce internal file system.

Each embedded HSM can access the per-release secrets for key derivation by first decrypting the master wrapping key, then using it to decrypt the remaining secrets.

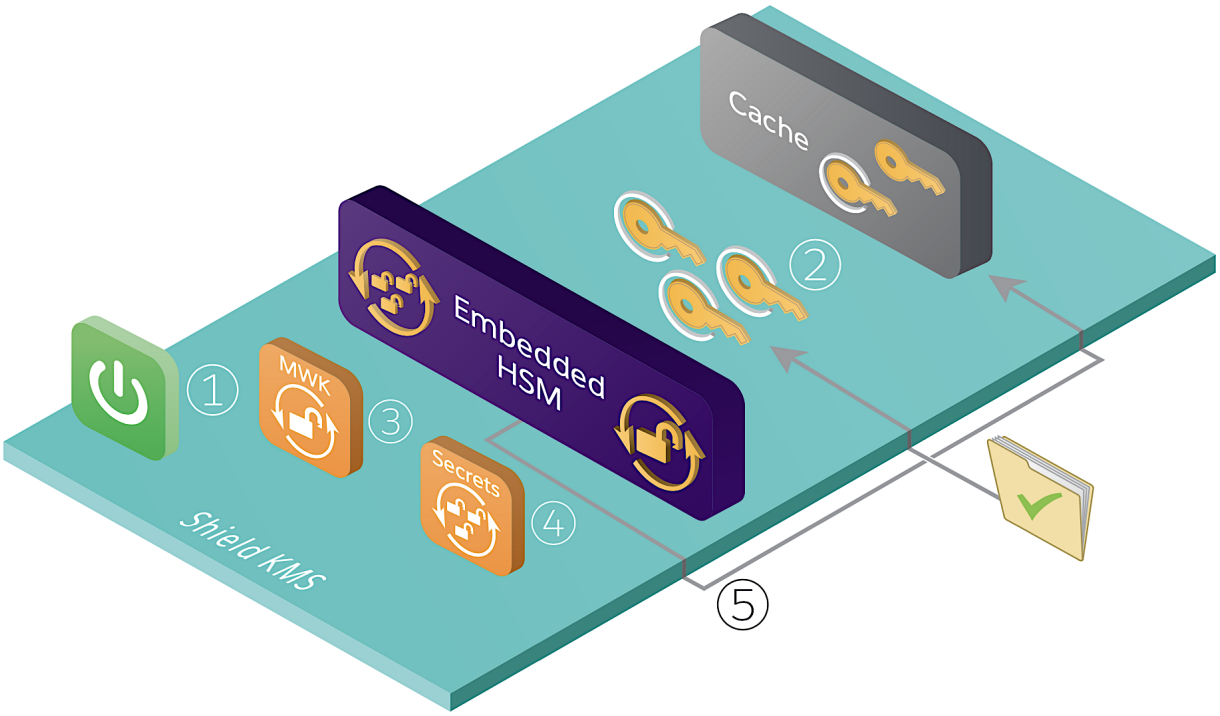


The process for exporting the secrets includes these steps:

1. The master wrapping key is read from the file system of the offline laptop and decrypted on the master HSM.
2. The master wrapping key is encrypted with each of the signed, embedded HSM's public encryption key.
3. The encrypted secrets and their hashes are checked into source control and stored in the Salesforce internal file system.

Shield KMS Startup

When the Shield KMS starts up in a production environment, it accesses each release's encrypted secrets stored in the internal file system. It then decrypts and validates them, and stores them in the encrypted key cache in preparation for deriving data encryption keys.

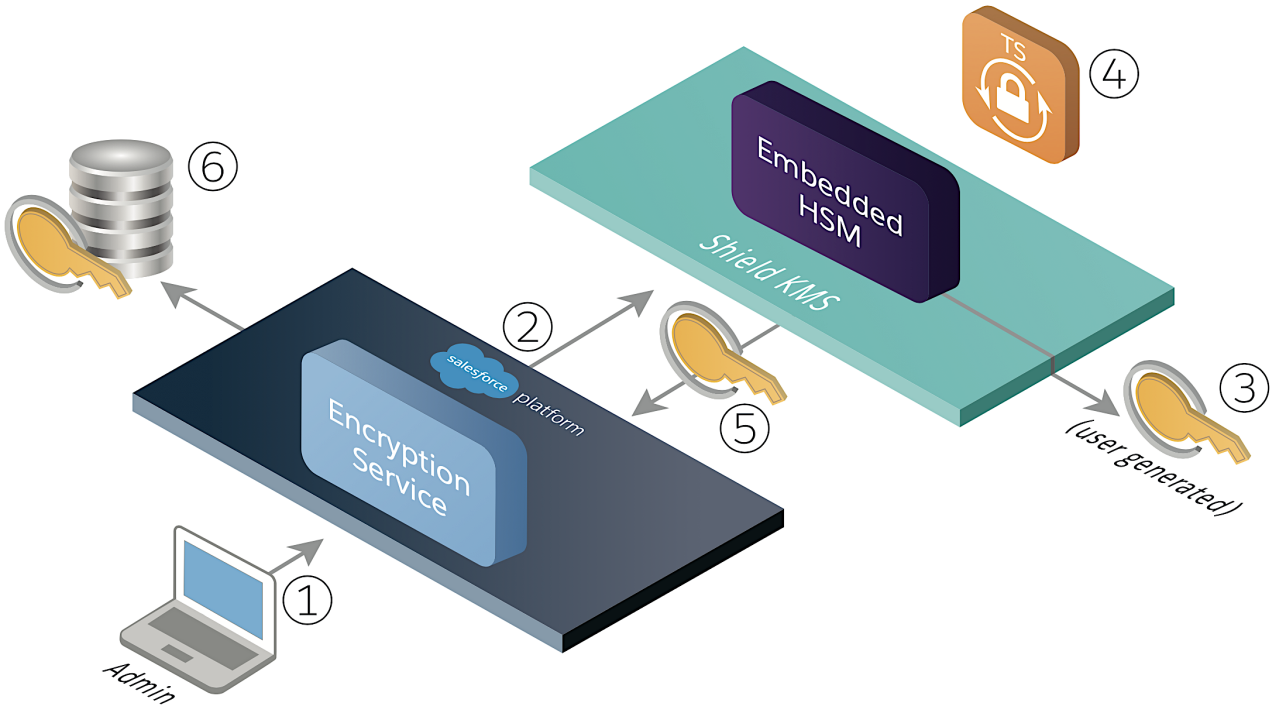


The process includes these steps:

1. The Shield KMS starts up.
2. The Shield KMS accesses the encrypted secrets and their hashes in the appropriate folders in the internal file system.
3. The embedded HSM decrypts the master wrapping key for each release.
4. Using the master wrapping keys, the Shield KMS decrypts the rest of the release secrets.
5. The Shield KMS validates all the keys and secrets against their hashes and then stores them in the encrypted key cache.

On-Demand Tenant Secret Generation

Customers can generate or upload key material every 24 hours in their production or Developer Edition orgs and every four hours in sandbox orgs. Key material can be destroyed at any time. When a customer generates new key material, all future data is encrypted with the final data encryption key. This final data encryption key is derived by default; customers can opt out of derivation and supply their own final data encryption key.



The on-demand tenant secret is generated by an embedded HSM connected to the Shield KMS.

The process of generating a tenant secret includes these steps:

1. An admin attempts to generate a new tenant secret using the UI or API.
2. The encryption service sends an authenticated request to the Shield KMS.
3. The embedded HSM generates the tenant secret (TS).
4. The Shield KMS encrypts the tenant secret with the per-release tenant wrapping key.
5. The Shield KMS sends the encrypted tenant secret back to the encryption service running on the Lightning Platform.
6. The encryption service stores the encrypted tenant secret securely in the database. The encrypted tenant secret is used to derive the org's specific data encryption keys on demand. This encrypted tenant secret can only be decrypted in the Shield KMS.

Customers Can Supply Their Own Key Material (BYOK)

Customers can supply their own tenant secret or final data encryption key using the Bring Your Own Key (BYOK) service. BYOK lets customers create tenant secrets and data encryption keys outside of Salesforce using their own crypto libraries, enterprise key management system, or hardware security module. Customers can encrypt their key material with a self- signed or certificate authority (CA) certificate's public key. They then grant Shield Platform Encryption's key management systems access to these keys, and can revoke access on demand via the Key Management tooling in Setup or programmatically via the API.

Once uploaded, customer-supplied tenant secrets work with the Salesforce key management systems just like Salesforce-generated tenant secrets. By default, when customer-supplied tenant secrets are uploaded, all subsequent data is encrypted with the key derived from the current master secret and the new customer-supplied tenant secret. This org-specific derived data encryption key is not persisted on disk.

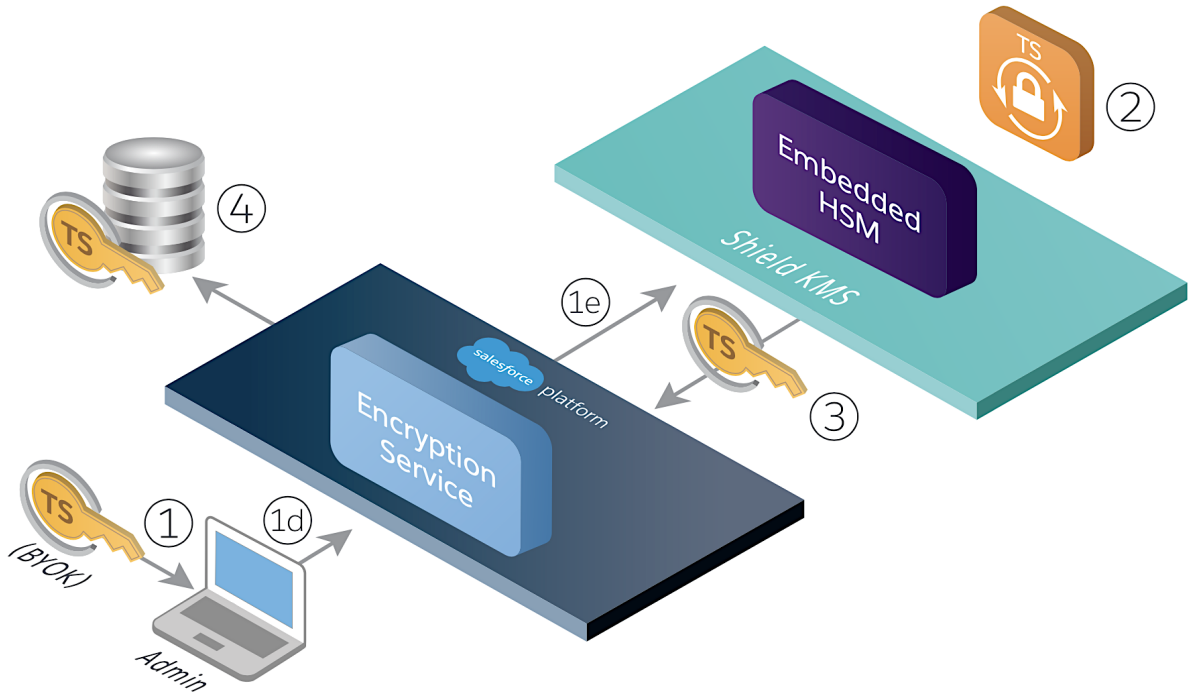
Customers can opt out of the key derivation process on a key-by-key basis. When customers opt out of derivation, they upload their own data encryption key, which is used to directly encrypt and decrypt data. These customer-supplied data encryption keys are wrapped with a tenant wrapping key before they are stored in the Salesforce database.

Alternatively, customers can create and store key material outside of Salesforce and use the Cache-Only Key Service to apply that key material to data in Salesforce. Customers can use an on-premises key service, host their own cloud-based key service, or use a cloud-based key brokering vendor. Keys are fetched on demand over a secure channel that the customer configures. These keys are wrapped with a cache key encryption key and placed in the encrypted key cache for encrypt and decrypt operations. Because cache-only keys bypass the key derivation process, they're used to directly encrypt and decrypt your data. Subsequent encryption and decryption requests go through the encrypted key cache until the cache-only key is revoked or rotated, or the cache is flushed. Once the cache is flushed, the Cache-Only Key Service fetches key material from your specified key service. The cache is regularly flushed every 72 hours, and certain Salesforce operations flush the cache on average every 24 hours. Destroying a data encryption key invalidates the corresponding data encryption key that's stored in the cache.

Customer-supplied key material can be uploaded once every 24 hours in production and Developer Edition orgs, and every four hours in sandbox orgs. They can be destroyed declaratively or programmatically by the customer at any time.

The process for generating and encrypting customer-supplied key material varies depending on whether customers use a crypto service, HSM, or key brokering service. However, all customer-supplied key material needs to meet the same basic requirements before it can be uploaded to Salesforce. Users need the Manage Encryption Keys permissions to upload and rotate key material and the Manage Certificates permission to manage certificates.

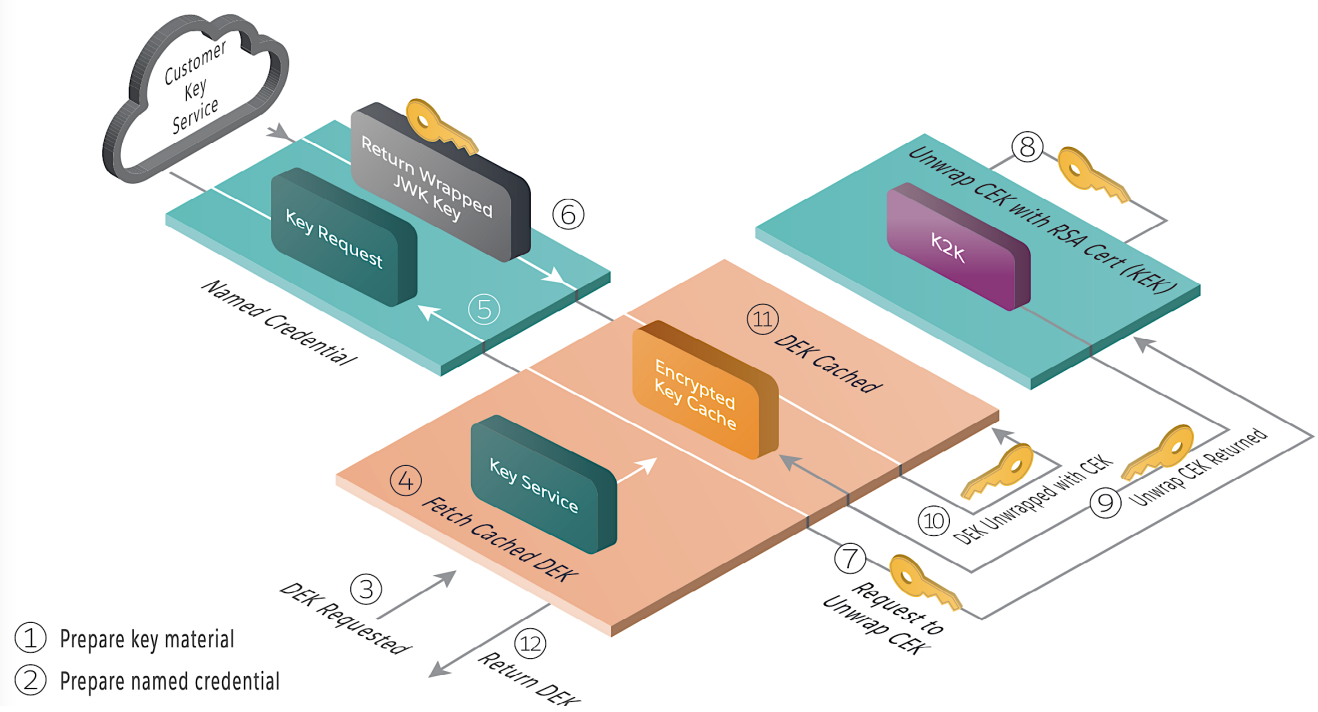
Customer-Supplied Tenant Secret Flow



1. Users prepare their tenant secret for upload.
   a. The user generates a BYOK-compatible certificate either declaratively or programmatically, where the certificate's private key is encrypted with an org-specific derived data encryption key. The user issues or creates a Certificate Signing Request (CSR), either self-signed or CA-signed. The user then downloads this CSR.

   b. The user generates a 256-bit tenant secret using the method of their choice, encrypts it with the public key from their BYOK-compatible CSR, and encodes the encrypted tenant secret to base64.
   c. The user calculates an SHA-256 hash of the plaintext tenant secret, then encodes this hash to base64.
   d. The user uploads both the encrypted tenant secret and hashed plaintext tenant secret files to Salesforce.
   e. The application server then passes the encrypted tenant secret and hashed plaintext tenant secret files to the Shield KMS.
   f. If customers don't opt out of derivation, the Shield KMS creates the BYOK-derived encryption key to unwrap the CSR's private key.
   g. The customer's uploaded tenant secret is decrypted using the BYOK CSR's private key.
   h. The tenant secret is then hashed using SHA-256, and compared to the SHA-256 hash provided by the customer.
2. If the hashes match, the Shield KMS encrypts the tenant secret with the per-release tenant wrapping key.
3. The Shield KMS sends the encrypted tenant secret back to the encryption service running on the Lightning Platform.
4. The encryption service stores the encrypted tenant secret securely in the database. The encrypted tenant secret is used to derive the org's specific data encryption keys on demand. This encrypted tenant secret can only be decrypted in the Shield KMS

Cache-Only Key Flow



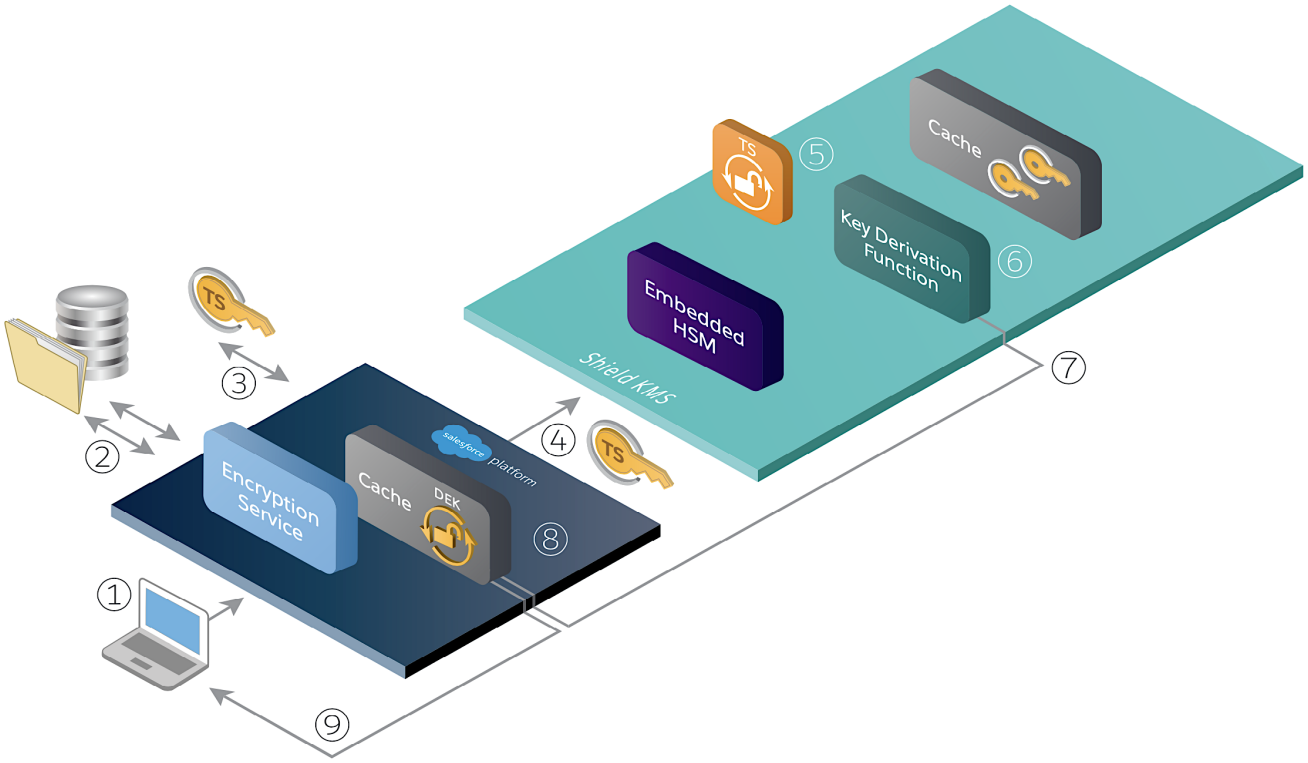
1. Users prepare their key material. Cache-only key material is wrapped in a JSON format.
   a. The user generates a 256-bit AES data encryption key using a cryptographically secure method.
   b. The user generates a 256-bit AES content encryption key using a cryptographically secure method.
   c. The user generates and downloads a BYOK-compatible certificate.

   d. The user creates a JWE protected header. The JWE protected header is a JSON object with three claims: the algorithm used to encrypt the content encryption key, the algorithm used to encrypt the data encryption key, and the unique ID of the cache-only key.
   e. The user encodes the JWE protected header as BASE64URL(UTF8(JWE Protected Header))
   f. The user encrypts the content encryption key with the public key from the BYOK certificate using the RSAES-OAEP algorithm. The user then encodes this encrypted content encryption key as BASE64URL(Encrypted CEK).
   g. The user generates an initialization vector for use as input to the data encryption key's AES wrapping, and then encodes it in base64url.
   h. The user wraps the data encryption key with the content encryption key.
      i. Encode the JWE header as ASCII(BASE64URL(UTF8(JWE Protected Header))).
      ii. Reform authenticated encryption on the data encryption key with the AES GCM algorithm. Use the content encryption key as the encryption key, the initialization vector (the bytes, not the base64URL encoded version), and the Additional Authenticated Data value, requesting a 128-bit Authentication Tag output.
      iii. Encode the resulting ciphertext as BASE64URL(Ciphertext).
      iv. Encode the Authentication Tag as BASE64URL(Authentication Tag).
   i. The user assembles the JWE as a compact serialization of all the preceding values, concatenating values separated by a period.

2. Users prepare a Salesforce named credential to serve as a secure callout connection, specifying their external key service as the endpoint.
3. The encryption service requests key material.
4. The encryption service queries the encrypted key cache for a data encryption key.
5. If no data encryption key is returned, the encryption service initiates a request to the endpoint specified in the customer-created named credential.
6. The customer's key service returns a content encryption key wrapped in a key encryption key, formatted as a JSON Web Encryption structure (JWE).
7. The encrypted key cache validates the response, and sends the content encryption key to the Shield KMS to be unwrapped.
8. The Shield KMS unwraps the content encryption key with the HSM-generated certificate's key encryption key.
9. The Shield KMS returns the unwrapped content encryption key to the encrypted key cache over a TLS-secured channel.
10. The data encryption key is unwrapped with the content encryption key.
11. The data encryption key is wrapped with a cache key encrypting key and placed in the encrypted key cache for encrypt and decrypt operations.

Encrypted Information Flow with Key Derivation

When attempting to read or write encrypted data, the encryption service transmits the request to the Shield KMS to retrieve the appropriate data encryption key (if the key isn't already in the encrypted key cache). The derived data encryption key is returned to the encryption service.



The process for deriving the data encryption key during a decrypt request includes these steps (note that encryption is nearly identical):

1. A user attempts to read encrypted data.
2. The Lightning Platform queries the data from the storage engine. This query could be for the database, search index, or file storage.
3. Based on metadata stored with the encrypted data, the encryption service retrieves the appropriate encrypted tenant secret from the database.
4. The encryption service sends an authenticated request for the derived key to the Shield KMS. The request includes the encrypted tenant secret.

   Data moving between the Shield KMS and the encryption service is encrypted by the TLS protocol, which uses a certificate signed by a dedicated Salesforce authority. This certificate's private key is stored locally in an encrypted form. The certificate's public and private keys are rotated regularly.
5. The Shield KMS decrypts the tenant secret with the appropriate tenant wrapping key in the encrypted key cache.
6. The Shield KMS derives the requested data encryption key using the appropriate master secret, master salt, and tenant secret as inputs to the key derivation function (PBKDF2WithHmacSHA256).
7. The Shield KMS sends the encrypted data encryption key back to the encryption service.
8. The encryption service decrypts the data encryption key. The data encryption key is encrypted with a cache key encryption key and stored in the encrypted key cache.
9. Using the data encryption key, the encryption service decrypts the customer data and returns it to the user.

PBKDF2 Inputs

Data encryption keys are derived using PBKDF2 with the following values as inputs.

- PRF—HmacSHA256

- Password—master secret XOR tenant secret
- Salt—master salt
- c—15,000
- dkLen—256

CHAPTER 4 Keys and Secrets

Cache Key Encrypting Key

Function

Org-specific key used to encrypt derived and customer-supplied data encryption keys in the encrypted key cache

Type

AES-256 key

How it's generated

Generated when a data encryption key is generated, rotated, or destroyed

Where it's stored

Encrypted with the tenant wrapping key and stored in the database

Content Encryption Key

Function

Unique key that wraps the data encryption key supplied by a customer's specified key service

Type

AES-256 key

How it's generated

Generated by customer

Where it's stored

As part of the customer-supplied data encryption key, encrypted with the cache encryption key and stored in the encrypted key cache

Data Encryption Key

Function

Org-specific key used to encrypt customer data (the "final" key)

Type

AES-256 key

How it's generated

Generated on the Shield KMS with PBKDF2, or generated by the customer

Where it's stored

Never persisted on disk in any form. Customer supplied, or derived on demand and stored in the encrypted key cache.

Embedded HSM Encryption Key Pair

Function

Used to encrypt and decrypt data that can only be accessed on the embedded HSM

Type

4096-bit RSA key pair

How it's generated

Generated once, upon initialization of embedded HSM

Where it's stored

Public key is signed by master HSM and stored in the Salesforce internal file system. Private key can't be accessed outside of embedded HSM.

Master HSM Encryption Key Pair

Function

Used to encrypt and decrypt data that can only be accessed on the master HSM

Type

4096-bit RSA key pair

How it's generated

Generated once, upon initialization of master HSM

Where it's stored

Public key is stored in the Salesforce internal file system. Private key can't be accessed outside of master HSM.

Master HSM Signing Key Pair

Function

Used to verify the public keys of embedded HSMs.

Type

4096-bit RSA key pair

How it's generated

Generated once, upon initialization of master HSM

Where it's stored

Signing key pair can't be accessed outside of master HSM

Master Salt

Function

Used as input to PBKDF2 to derive data encryption keys.

Type

256-bit value

How it's generated

Generated once each release by the master HSM

Where it's stored

Encrypted with master wrapping key and stored in the Salesforce internal file system

Master Secret

Function

Used in conjunction with organization tenant secrets to derive data encryption keys.

Type

256-bit value

How it's generated

Generated once each release by the master HSM

Where it's stored

Encrypted with master wrapping key and stored in the Salesforce internal file system.

Master Wrapping Key

Function

Used to encrypt the master secret, master salt, tenant wrapping key, and transit wrapping private key before they're stored in the Salesforce internal file system

Type

AES-256 key

How it's generated

Generated once each release by the master HSM

Where it's stored

Encrypted with each embedded HSM's public encryption key and the master HSM's public encryption key and stored in the Salesforce internal file system

Search Index IV

Function

Used as input to PBKDF2 to derive data encryption keys

Type

256-bit value

How it's generated
Generated per search index segment when the search index segment is updated

Where it's stored
Encrypted with search key and stored in the search index segment header

Search Index Data Encryption Key

Function
Used to encrypt and decrypt the search index segment after indexing completes

Type
AES-256 key

How it's generated
Generated on the Shield KMS with PBKDF2

Where it's stored
Never persisted on disk in any form. Derived on demand and stored in the encrypted key cache.

Tenant Secret

Function
Combined with the master secret to derive a unique data encryption key.

Type
256-bit value

How it's generated
Generated on customer demand by an embedded HSM on the Shield KMS

Where it's stored
Encrypted with the tenant wrapping key, sent from the Shield KMS to an application server on the Lightning Platform, and stored in the database

Tenant Wrapping Key

Function
Used to encrypt tenant secrets before they're stored in the database

Type
AES-256 key

How it's generated
Generated once each release by the master HSM

Where it's stored
Encrypted with the master wrapping key and stored in the Salesforce internal file system

SEE ALSO:

Get Started with Shield Platform Encryption

Use Trailhead to learn how you can go the extra mile and encrypt your data