salesforce platform

Shield Platform Encryption Architecture

Strengthen your data's security while controlling the lifecycle of your keys

Spring '18

Contents

WHY ENCRYPT?

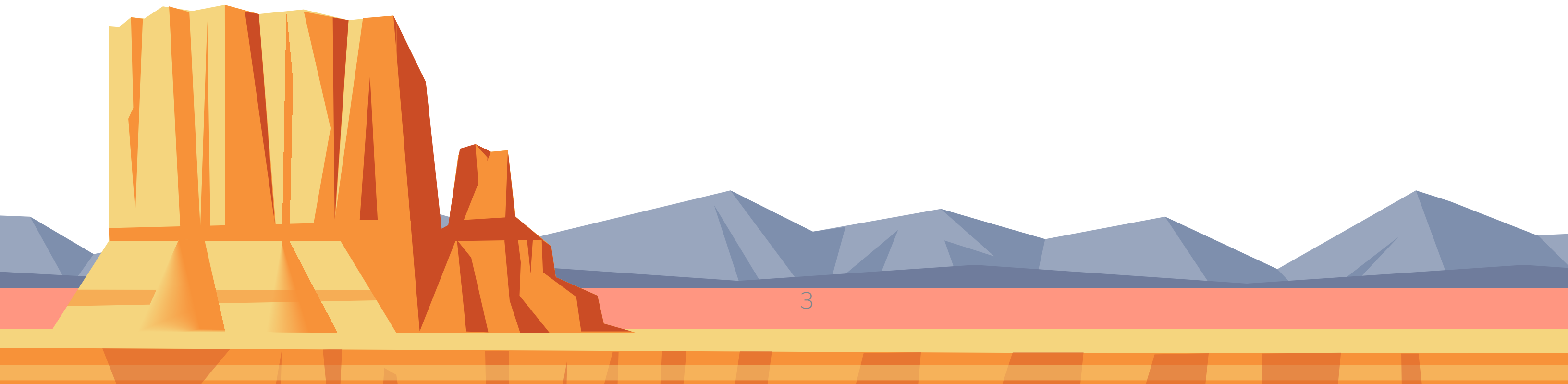
The need for encryption

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

The Verizon 2017 Data Breach Investigations Report counted over 42,000 security incidents and nearly 2,000 confirmed data breaches worldwide in 2016. A study of 1,792 data breach incidents in 2016 showed that encryption was used to secure breached data in only 4.2% of cases. According to some estimates, over 9 billion records have been compromised since 2013.

New security vulnerabilities such as Heartbleed, Shellshock, POODLE, and Spectre are reported on a regular basis. 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 is 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 Lightning Platform to build applications that require PII and other sensitive, confidential, or proprietary data. With this sensitive data stored on the Lightning Platform, customers want additional layers of protection on top of our standard security measures. Extra features such as authentication and single sign-on, granular access controls, and advanced activity monitoring give customers control over when and how they protect their data.



Balancing 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.

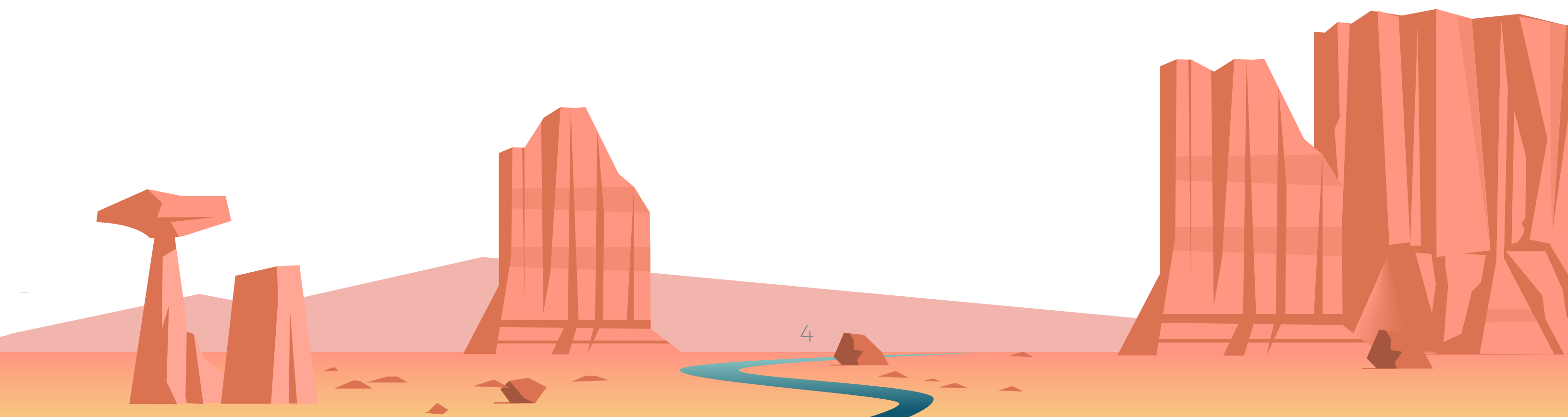
This pragmatic approach includes three requirements shared by a wide variety of customers in regulated industries such as financial services, healthcare, and life sciences, as well as manufacturing, technology, and government.

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 Lightning Platform functionality.

However, there's a tradeoff between strong security and functionality. If data is encrypted at rest, preserving Salesforce functionality becomes difficult, if not impossible, depending 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.



WHY ENCRYPT?

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. We don't address data residency or remote key management, which require off-Salesforce solutions and typically involve on-premises software and complex integrations. To encrypt data at rest and preserve functionality, we built the encryption services natively into the Lightning Platform.

NATIVELY INTEGRATE ENCRYPTION AT REST WITH KEY SALESFORCE FEATURES.

One of the things that makes the Lightning Platform so remarkable is that it is 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 this a reasonable tradeoff in favor of security. However, we recognize that in some cases, business requirements depend on preserving more functionality, which might influence what data customers decide to encrypt. In those cases, we offer deterministic encryption on a beta basis. Deterministic encryption uses the Advanced Encryption Standard (AES) with 256-bit keys using CBC mode and a customer- and field-specific static IV. In this way, deterministic encryption only decreases encryption strength as much as is minimally necessary to allow filtering.

EMPOWER CUSTOMERS TO DRIVE THE KEY LIFECYCLE.

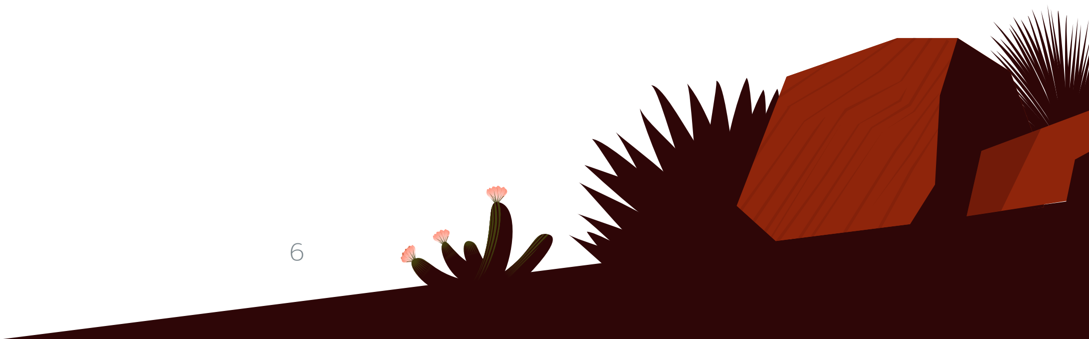
We built a key management framework that scales to our multitenant model and gives you complete control over the key management lifecycle. Since the encryption service is built natively into the Lightning Platform, the encryption keys must reside in the Salesforce environment. Adhering to the principle that customers should 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 tenant secrets and data encryption keys 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 are 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, and attachments. 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 are most concerned about protecting against end-user or administrative account takeover attacks, which are usually achieved through social engineering and malware infection, data encryption may not be an appropriate control against such a threat. Consider instead malware detection and activity monitoring as ways to identify when users may 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.

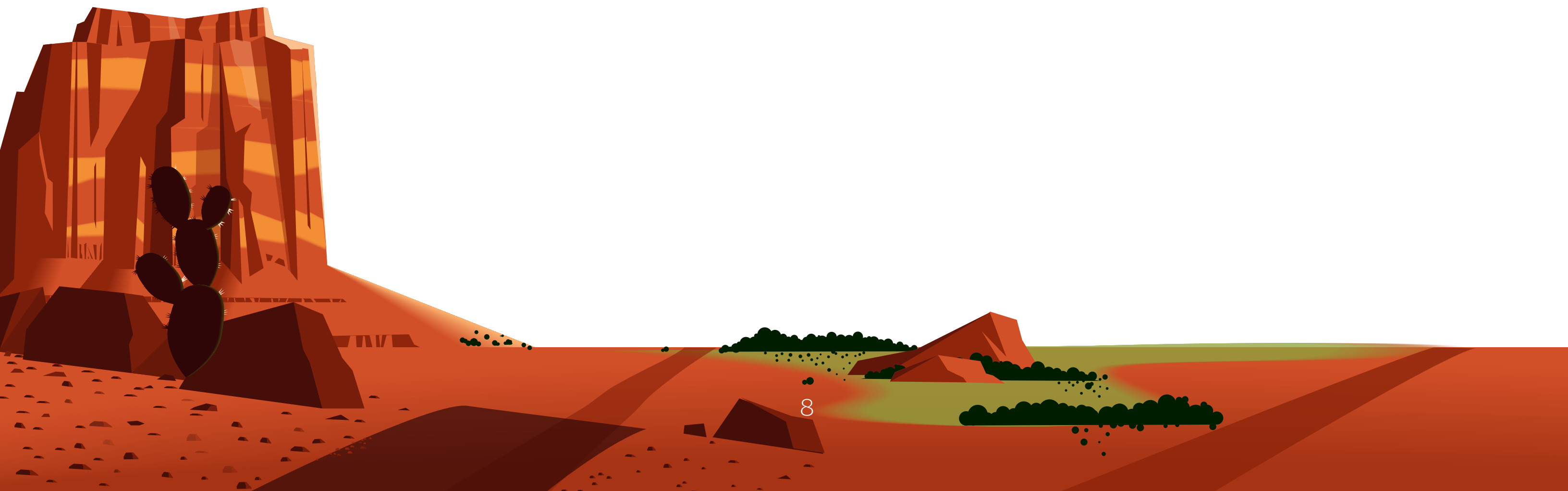
After completing a threat modeling exercise, use the outcome to inform a granular data classification. (You may want to keep the Salesforce Security Guide handy as you work through this project.) Identify data elements that are sensitive, private, or confidential. Your resulting strategy should be to encrypt only the most sensitive of those data elements. This will help to balance stronger data protection controls against the need to build and preserve critical business functionality on the Lightning Platform or when using Sales Cloud or Service Cloud.

WHY ENCRYPT?

What gets encrypted?

In contrast to Classic Encryption, which uses a custom field type in the Salesforce data model, Shield Platform Encryption allows you to encrypt a variety of standard fields, custom fields, and files. We use metadata to keep information in these files and fields secure while preserving the ability to perform common business tasks.

We make more fields, files, and data elements available for encryption every year. For the complete list of fields and files you can encrypt with Shield Platform Encryption, see Encrypt Fields, Files, and Other Data Elements in Salesforce Help.



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

Feature	Classic Encryption	Shield Platform Encryption
Pricing	Included in base user license	Additional fee applies
Encryption at Rest	✓	✓
Native Solution (No Hardware or Software Is 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		✓

How Shield Platform Encryption Works

ENCRYPTION IS BUILT RIGHT INTO THE PLATFORM

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. This ensures that encrypted data isn't persisted 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, and the application's code base can be patched or upgraded without breaking tenant-specific customizations. See the Lightning Platform Fundamentals: A 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. In the case of decryption, data is decrypted as it's materialized. It is then pushed up through the application pipeline and appears in plaintext to the user who requested it.

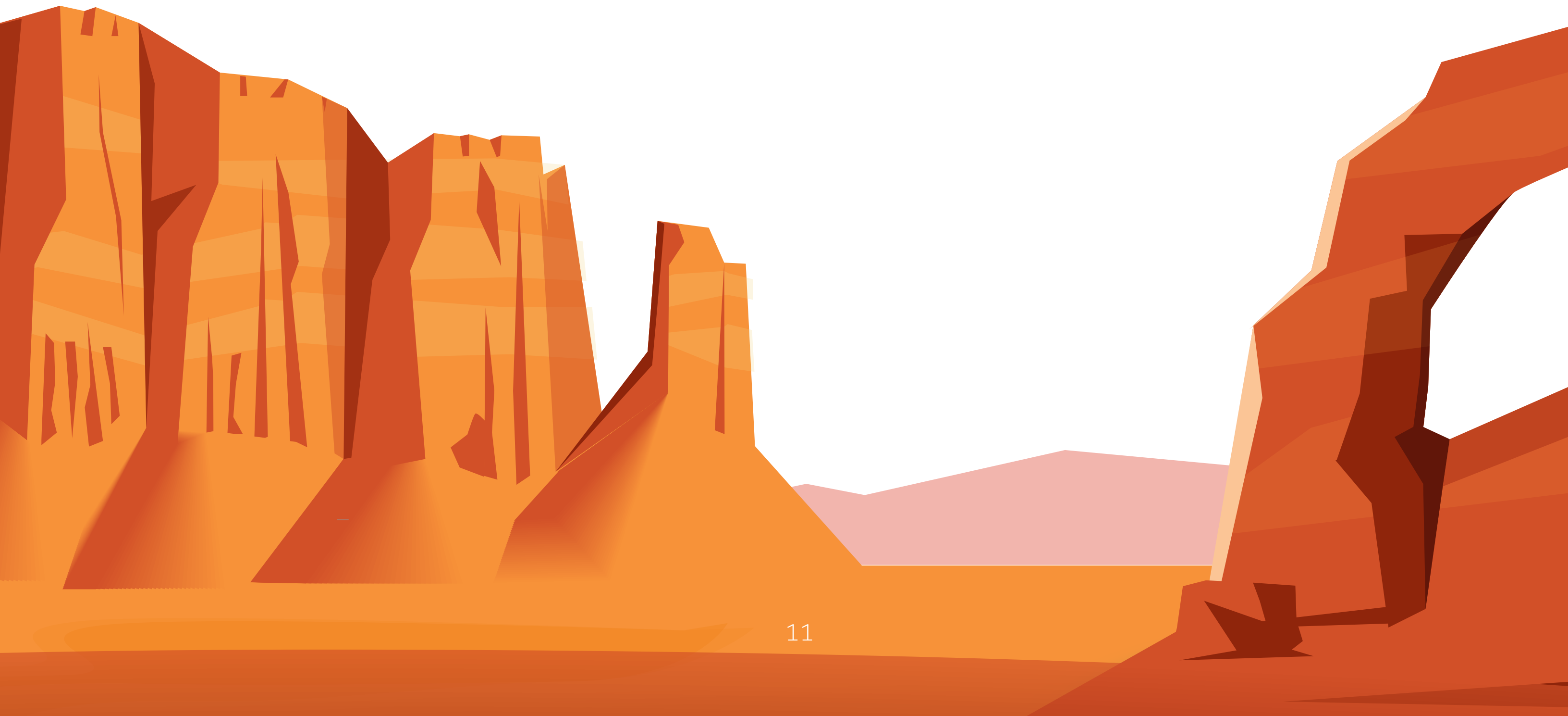
By embedding the encryption metadata in the UDD, the Shield Platform Encryption architecture allows customers to choose what data to encrypt. Performing the encryption work in Shield Platform Encryption's application layer enables the engine to strictly manage the flow of encrypted data from the application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

CRYPTOGRAPHIC LIBRARY AND ALGORITHMS

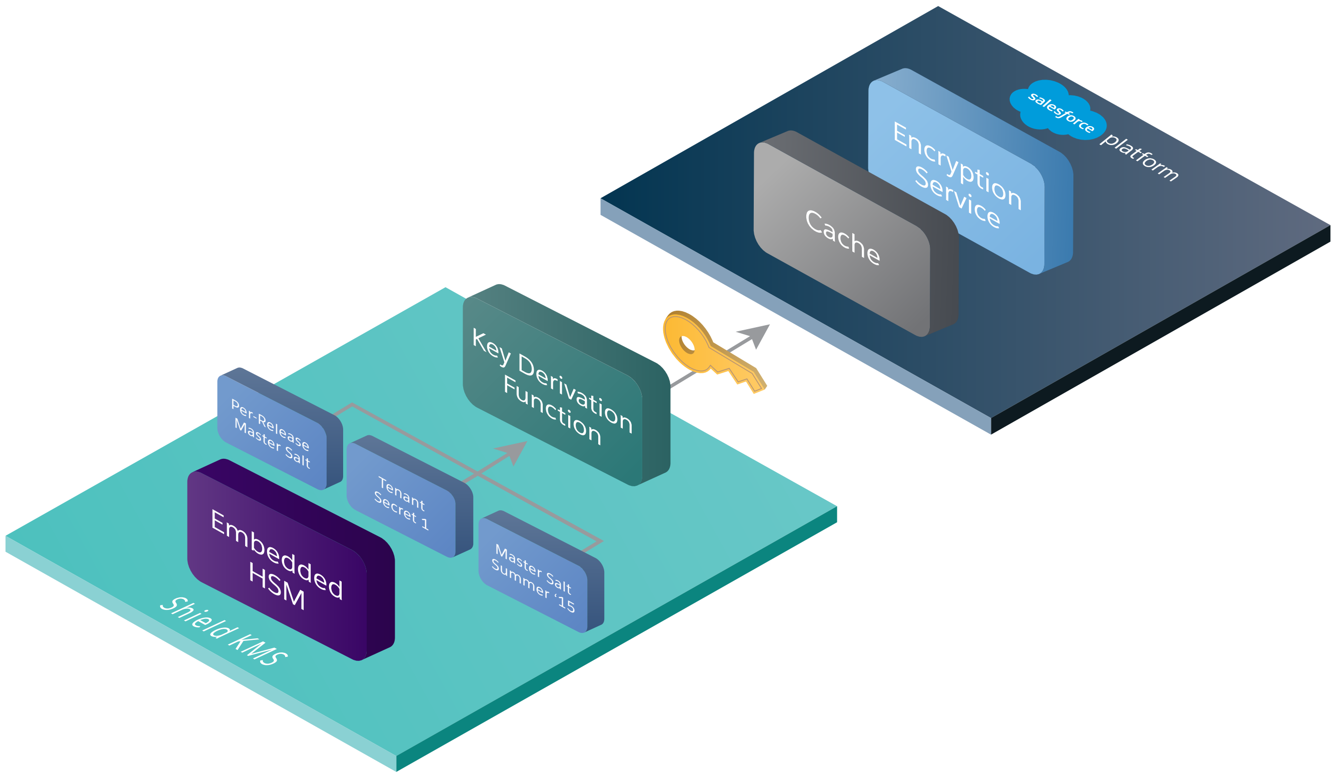
Shield Platform Encryption uses the Java Cryptography Extension (JCE) to encrypt and decrypt data. Specifically, Shield Platform Encryption uses the Advanced Encryption Standard (AES-256) in CBC mode with a random IV.

ENCRYPTED KEY CACHE

Salesforce has taken steps to reduce access to encryption key material across the servers where encrypted keys might be stored. Rather than storing data encryption keys on the application server, they are stored in an encrypted key cache. This 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 period of time to perform both encryption and decryption operations. Once 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 and stored encrypted in the database. 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 (encrypted) back to the encryption service running on the Lightning Platform and stored in the encrypted key cache.

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

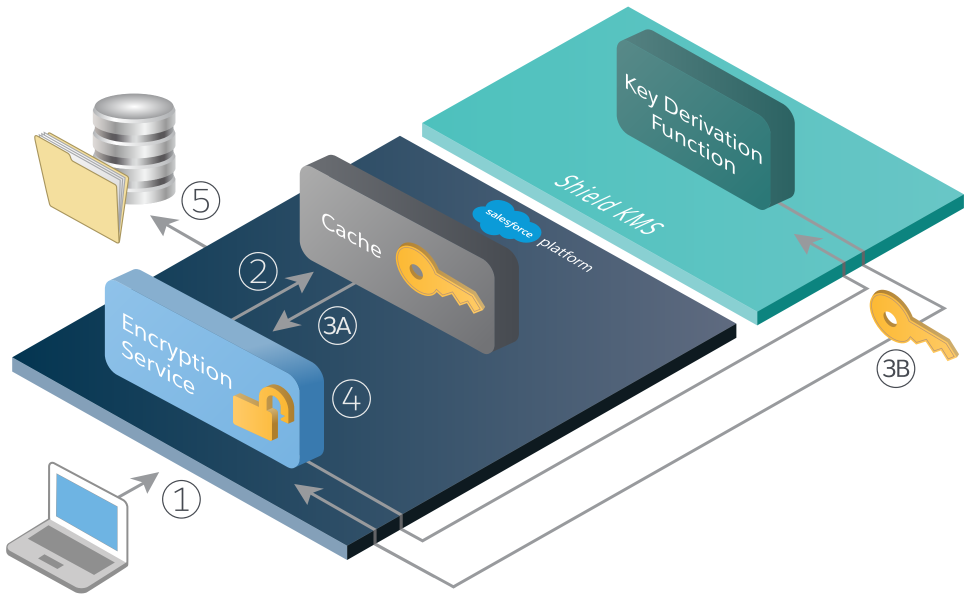
STORING ENCRYPTED PAYLOADS

Encrypted data is stored in the database with its metadata. The metadata includes:

- A bit that indicates that 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, 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, which 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 nondeterministically encrypt the data.

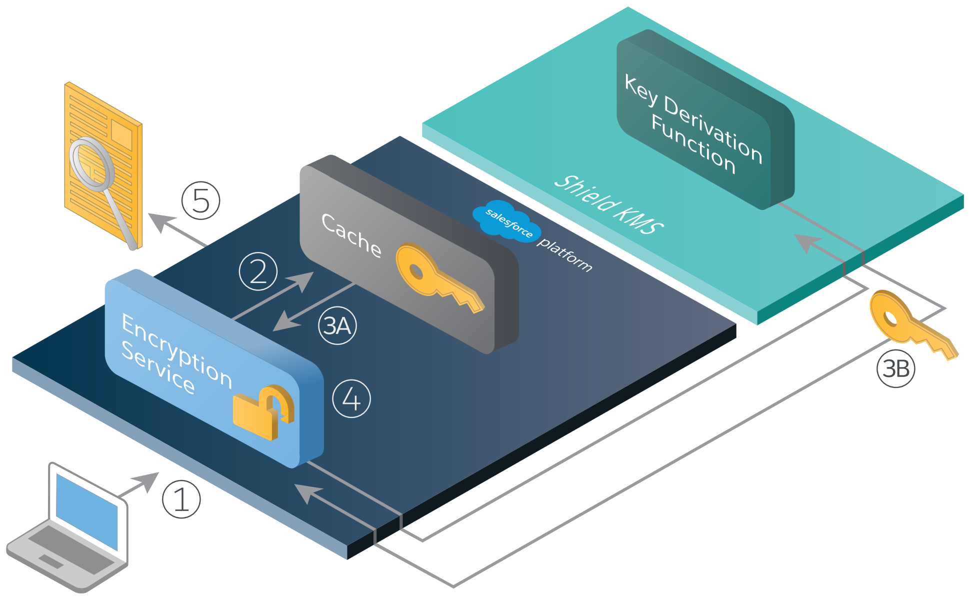
Shield Platform Encryption process flow



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 should be encrypted before storing it in the database.
2. If so, 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 so, the encryption service retrieves the key.
   b. Otherwise, the service sends a derivation request to the Shield KMS and returns it to the encryption service running on the Lightning Platform.
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 to determine which data elements need to 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

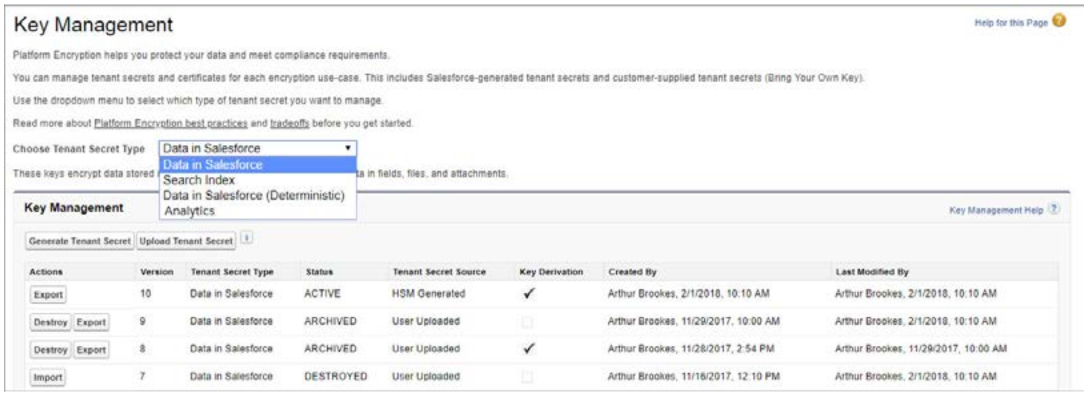
1. The core application determines if the search index segment should be encrypted or not based on metadata.
2. If the search index segment should be encrypted, the encryption service checks for the matching search encryption key ID in the encrypted key cache.
3. The encryption service determines if 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. Otherwise, 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.
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 search index encryption process above.
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


Key Management

Help for this Page

Platform Encryption helps you protect your data and meet compliance requirements.

You can manage tenant secrets and certificates for each encryption use-case. This includes Salesforce-generated tenant secrets and customer-supplied tenant secrets (Bring Your Own Key).

Use the dropdown menu to select which type of tenant secret you want to manage.

Read more about Platform Encryption best practices and tradeoffs before you get started.

Choose Tenant Secret Type: Data in Salesforce

These keys encrypt data stored ta in fields, files, and attachments.

Key Management

Key Management Help

Actions	Version	Tenant Secret Type	Status	Tenant Secret Source	Key Derivation	Created By	Last Modified By
Export	10	Data in Salesforce	ACTIVE	HSM Generated	✓	Arthur Brookes, 2/1/2018, 10:10 AM	Arthur Brookes, 2/1/2018, 10:10 AM
Destroy Export	9	Data in Salesforce	ARCHIVED	User Uploaded	☐	Arthur Brookes, 11/29/2017, 10:00 AM	Arthur Brookes, 2/1/2018, 10:10 AM
Destroy Export	8	Data in Salesforce	ARCHIVED	User Uploaded	✓	Arthur Brookes, 11/28/2017, 2:54 PM	Arthur Brookes, 11/29/2017, 10:00 AM
Import	7	Data in Salesforce	DESTROYED	User Uploaded	☐	Arthur Brookes, 11/18/2017, 12:10 PM	Arthur Brookes, 2/1/2018, 10:10 AM

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 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. 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.

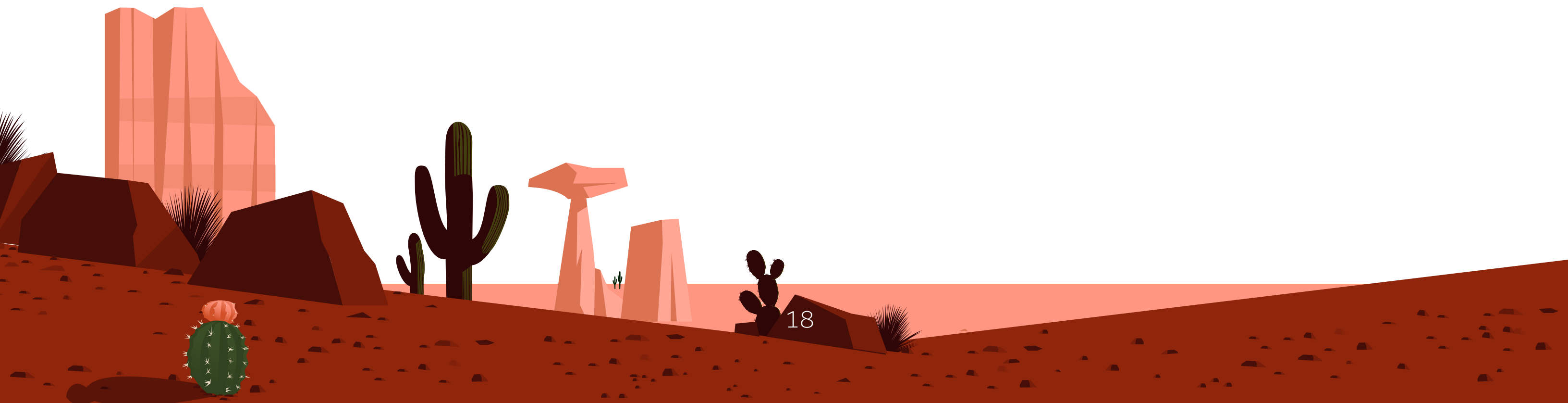
It's possible to have more than one active tenant secret or data encryption key in an org. You can apply specific keys to data stored in different areas of Salesforce. For example, search index files are stored separately from other Salesforce data, so customers can apply key material to specific data in those files. Only the most recent tenant secret or data encryption key of a given type is active, meaning 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. Once you destroy key material, it's fully removed from the database 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. Existing data stays encrypted with the former key material.

Salesforce can run a background encryption process 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.



Key Derivation Architecture

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

These secrets and secret-wrapping keys are initialized by a master HSM at the start of each release, or in the case of customer-driven tenant secrets, 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. These customer-supplied data encryption keys are used to directly encrypt and decrypt data. This 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 encyrpted key cache in preparation for key derivation.

TWO OPTIONS FOR PROVIDING KEY MATERIAL:

- **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.

- **CUSTOMER-SUPPLIED TENANT SECRET OR DATA ENCRYPTION KEY UPLOAD**
  The Salesforce Shield Bring Your Own Key (BYOK) service allows customers to create tenant secrets and data encryption keys outside of Salesforce using the customer's crypto libraries, enterprise key management system, or hardware security module. They then grant Shield Platform Encryption's key management machinery access to these keys. Customers can encrypt their key material with a self-signed or certificate authority (CA) certificate's public key. They can revoke Salesforce's access to this key material on demand via the Key Management tooling in Setup or programmatically via the API.

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 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

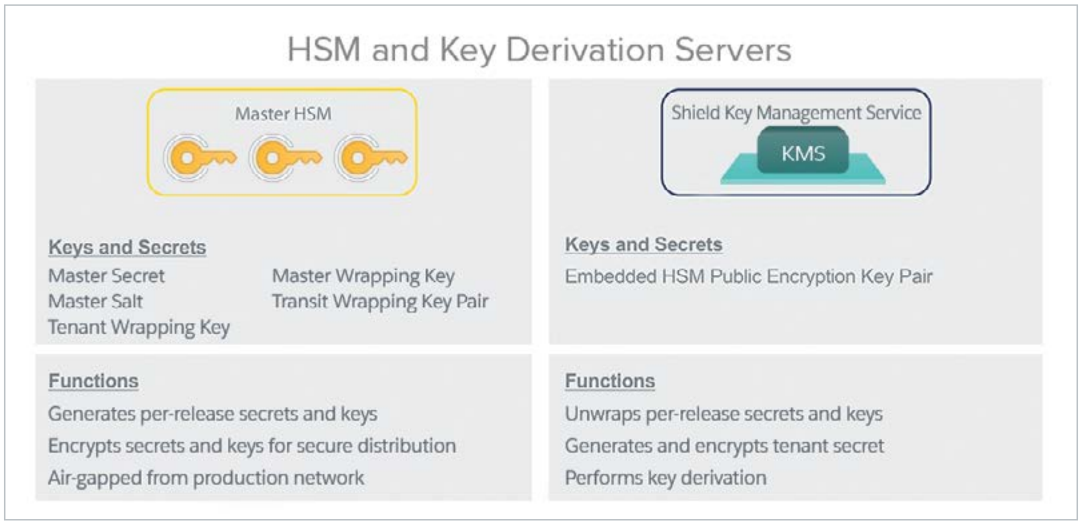
Clusters of load-balanced servers deployed to Salesforce's production data centers that derive data encryption keys from master secrets and tenant secrets.

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 and Key Derivation Servers
Master HSM
Shield Key Management Service
KMS
Keys and Secrets
Master Secret
Master Salt
Tenant Wrapping Key
Master Wrapping Key
Transit Wrapping Key Pair
Keys and Secrets
Embedded HSM Public Encryption Key Pair
Functions
Generates per-release secrets and keys
Encrypts secrets and keys for secure distribution
Air-gapped from production network
Functions
Unwraps per-release secrets and keys
Generates and encrypts tenant secret
Performs key derivation

HSM Initialization

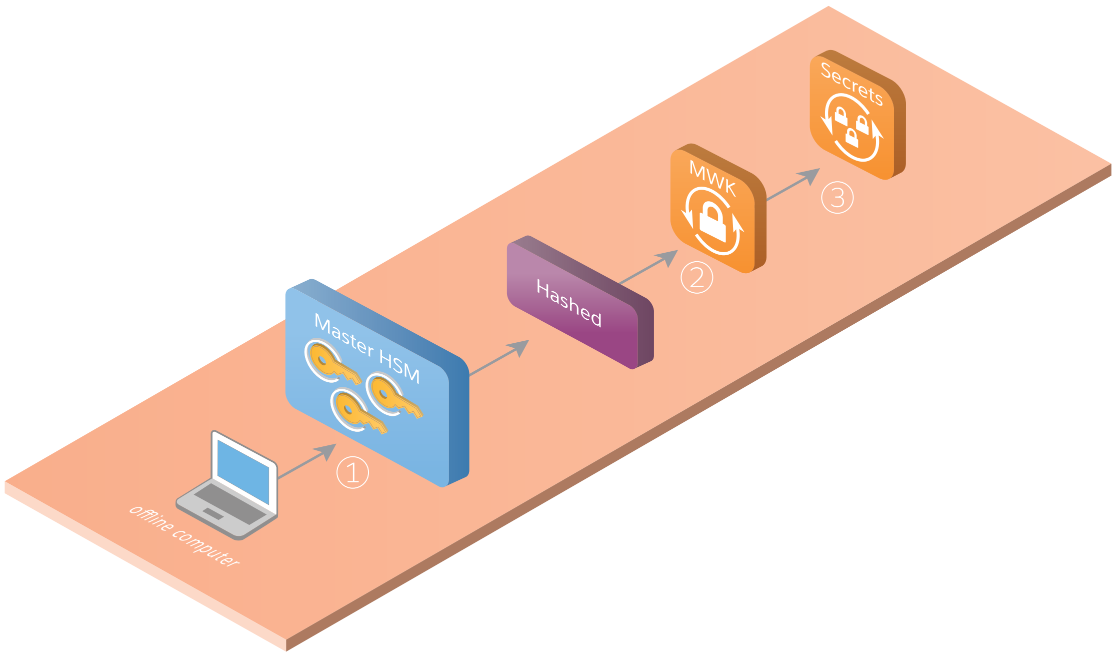
The master HSM and the embedded HSMs must be initialized before they're used. For the master HSM, initialization means creating a master HSM encryption key pair and a master HSM signing key pair. For each embedded HSM, it means creating an embedded HSM encryption key pair.

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, which 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.

KEY DERIVATION ARCHITECTURE

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

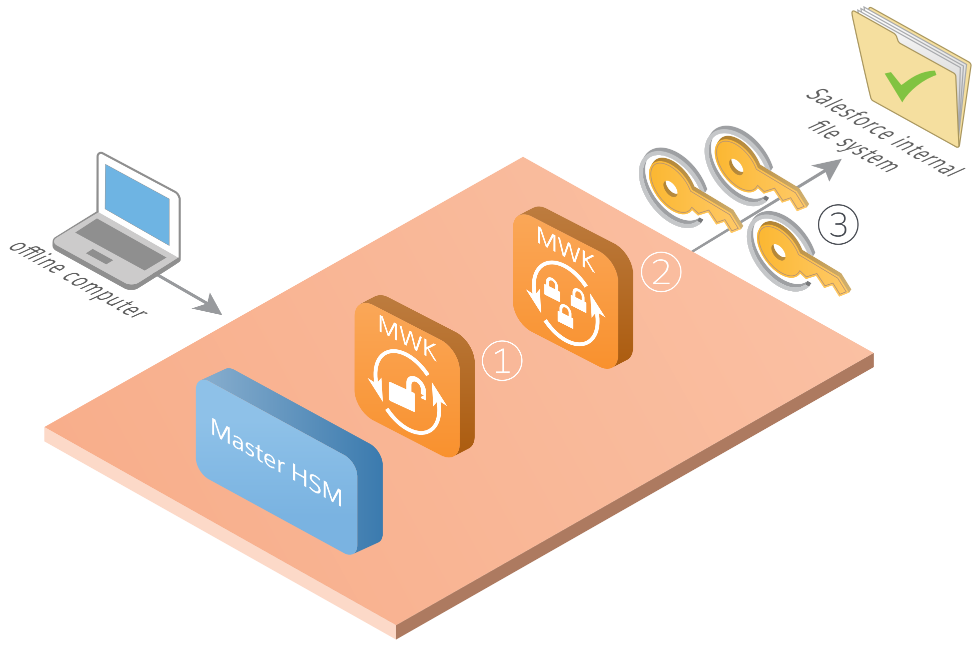
    · Transit wrapping key pair

Each secret is hashed using SHA-256. For definitions of each secret and key, refer to the Key and Secret Glossary.

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



Once all the secrets are hashed and encrypted, they are 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.

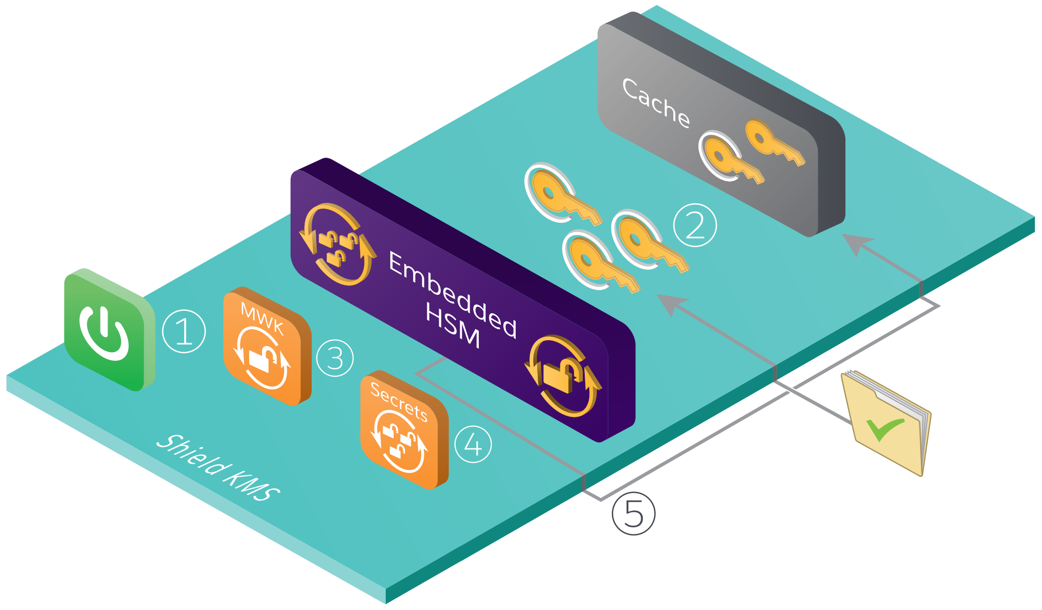
The process for exporting the secrets includes these steps:

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 HSMs 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

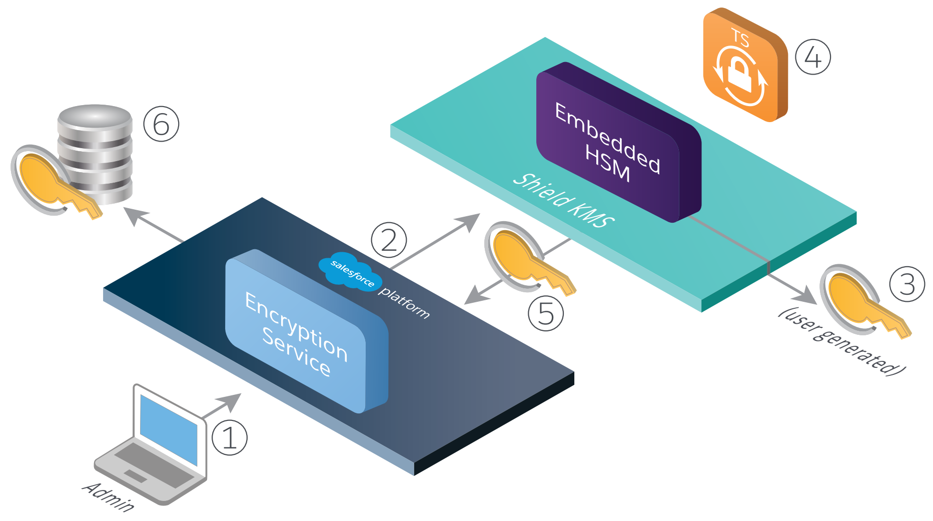


When the Shield KMS starts up in a production environment, it accesses each release's encrypted secrets stored in the internal file system, decrypts and validates them, and stores them in the encrypted key cache in preparation for deriving data encryption keys.

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.

KEY DERIVATION ARCHITECTURE

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. Once uploaded, customer-supplied tenant secrets work with the Salesforce key management machinery 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.

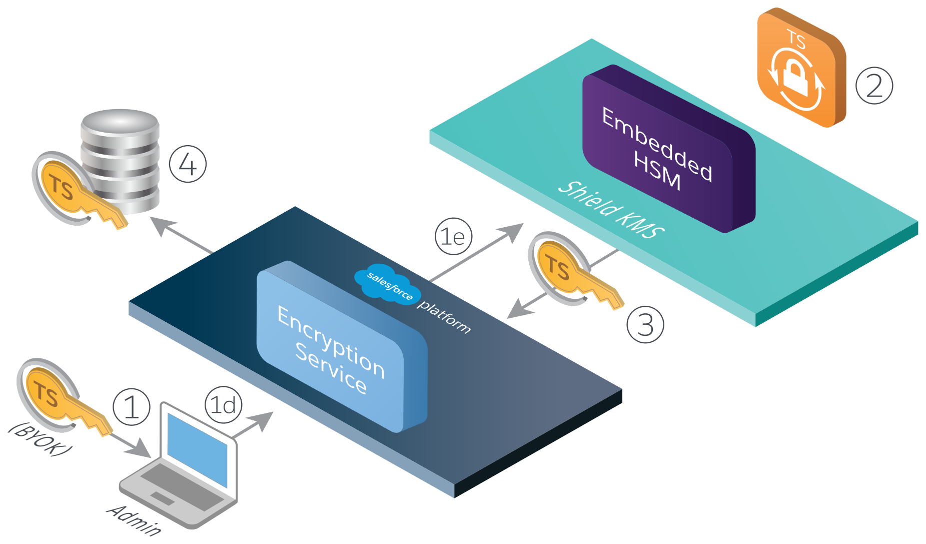
Customers can opt out of the key derivation process on a key-by-key basis. When customers opt out of derivation, they upload their own data encryption key, which is used to directly encrypt and decrypt data. These customer-supplied data encryption keys are wrapped with a tenant wrapping key before they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

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. Grant these permissions to authorized users only.

KEY DERIVATION ARCHITECTURE

Customer-Supplied Tenant Secret Flow



1. Users prepare their tenant secret for upload.

   a. The user generates BYOK-compatible certificate either declaratively or programmatically, where the certificates 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. 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.

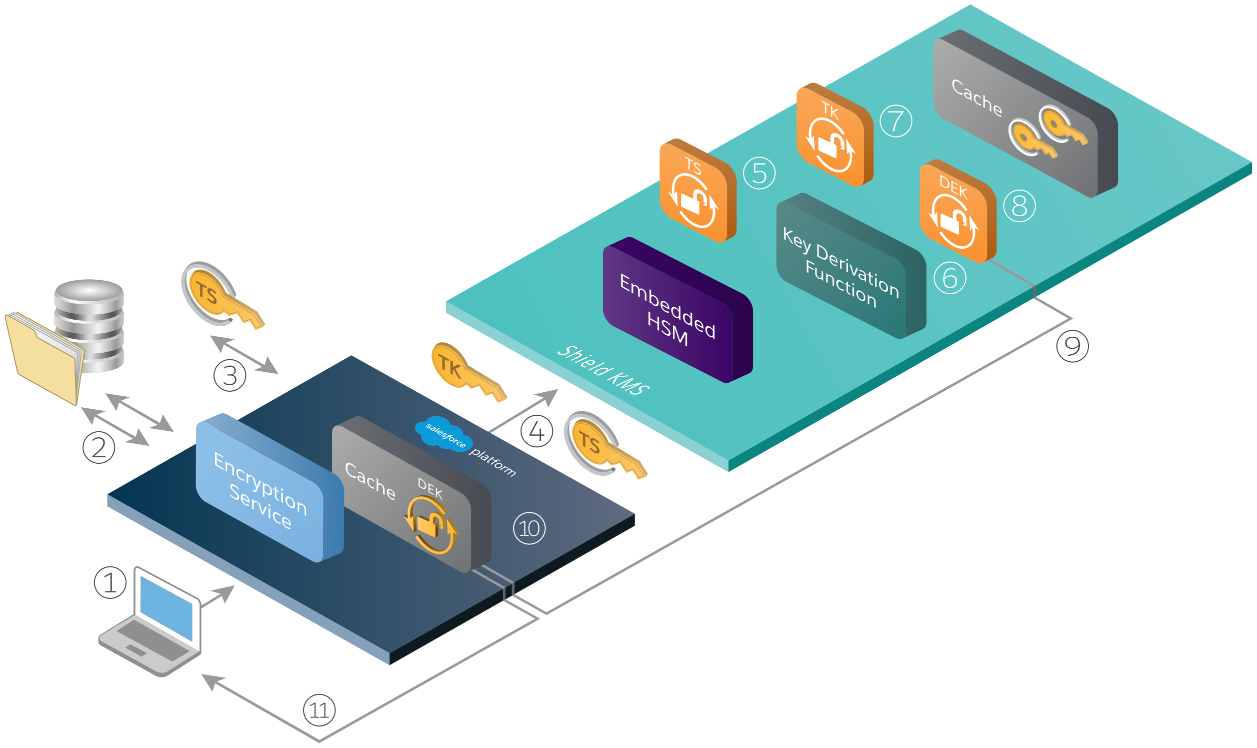
2. If the hashes match, the Shield KMS encrypts the tenant secret with the per-release tenant wrapping key.

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.

KEY DERIVATION ARCHITECTURE

Encrypted Information Flow with Key Derivation



When a customer attempts to read or write encrypted data, the encryption service transmits the request to the Shield KMS to retrieve the appropriate data encryption key (unless the key is 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 could be 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 following information:

   a. The encrypted tenant secret.

   b. A unique transit key (TK) that's generated on the Lightning Platform application server each time it boots up.

   The transit key is used to encrypt the derived data encryption key before it's sent back to the encryption service. The transit key is itself encrypted by the encryption service with the transit wrapping public key, which is half of the transit wrapping key pair generated by the master HSM each release.

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 decrypts the application server's transit key with the transit wrapping private key.

8. The Shield KMS encrypts the data encryption key (DEK) with the transit key. This ensures that the data encryption key isn't transmitted in the clear.

9. The Shield KMS sends the encrypted data encryption key back to the encryption service.

10. 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.

11. Using the data encryption key, the encryption service decrypts the customer data and returns it to the user.

PBKDF2 INPUTS

By default, 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

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

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
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 cannot 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 cannot 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 cannot 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 are 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 are 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

TRANSIT KEY

FUNCTION	Used to encrypt data encryption keys before they are sent from the Shield KMS to the application server on the Lightning Platform
TYPE	AES-256 key
HOW IT'S GENERATED	Generated on each application server upon startup and sent to the key derivation server upon a key derivation request
WHERE IT'S STORED	Stored in application server memory. Never persisted on disk.

TRANSIT WRAPPING KEY PAIR

FUNCTION	Used to encrypt the transit key before it's sent from the application server to the Shield KMS
TYPE	4096-bit RSA key pair
HOW IT'S GENERATED	Generated once each release by the master HSM
WHERE IT'S STORED	The public key is stored in the Salesforce internal file system. The private key is encrypted with master wrapping key and stored on the Salesforce internal file system

Next Steps

GET STARTED WITH SHIELD PLATFORM ENCRYPTION

To see how Shield Platform Encryption can help your company, contact your account executive or call 1-844-463-0828 today.

LEARN MORE

LEARN HOW YOU CAN GO THE EXTRA MILE AND ENCRYPT YOUR DATA ON TRAILHEAD.

LEARN MORE