Encryption at Rest for Extracts in Tableau Server 2019.3

Introduction

This whitepaper provides a deep dive into the encryption at rest for extracts feature introduced in Tableau Server 2019.3. Encryption at rest for extracts gives customers the ability to specify additional protection for Tableau data extract files persisted on Tableau Server. It allows for Operating System users to be created that can access the Server file system, but lack access to the specific encryption keys that are needed to decrypt the extract files stored on them. The solution provides a simple, secure, and performant implementation that can be used on its own or in addition to external volume-based encryption solutions.

Contents

What is encrypted?

The encryption at rest feature encrypts Tableau extracts at rest on Tableau Server. Extracts are long-term persisted data files containing data from customer data sources.

- This feature is only available in Tableau Server. Extracts in other products such as Tableau Desktop and Tableau Prep are not encrypted. Tableau Online currently uses volume-based encryption.
- Extracts are decrypted into memory prior to being queried by Tableau
- Extracts are automatically decrypted temporarily on Server before an authorized user downloads them from Tableau Server.
- Only Hyper extracts (.hyper files) are encrypted. Legacy extracts (.tde files) need to be updated to Hyper extracts to be encrypted. This will happen seamlessly in the background so it is transparent to the end-user.

What isn't encrypted?

- Temporary files and cache files are not encrypted at rest with this feature. These files may contain row-level data from the database but will be automatically removed when they are no longer needed. Tableau Server allows you to control certain caching settings, although doing so has a performance trade-off. Refer to the product documentation for more information about cache settings.
- Metadata such as Workbooks files (.twb) and data source files (.tds) are not encrypted with this feature. These files contain data describing the data such a database table column names and formatting instructions, and may also contain some row level data such as filter lists and dashboard thumbnail images.
- Other non-extract data files, such as Excel or JSON files, are not encrypted with this feature unless they are published as extracts.

Extract encryption provides protection above what full disk encryption provides by protecting the extracted information from disclosure if the extract file is accessed. Encrypted extracts cannot be decrypted without access to a master key. The key management process is described in a subsequent section.

Enabling encryption on Tableau Server

Encryption settings are controlled at the site level. A site can be configured with one of three settings: Disabled, Enabled, or Enforced.

- **Disabled:** If encryption is set to Disabled (the default), then no extracts will be encrypted.
- **Enabled:** If encryption is set to Enabled, then extracts are encrypted upon specification by the publisher, owner, or server admin.
- **Enforced:** If encryption is set to Enforced, all extracts will be encrypted in the site, including existing and newly published ones.

When does encryption happen?

When the state of an extract is changed to encrypt, a background job is created to encrypt the extract. There is no interruption to users during this process and the unencrypted extract will only be removed by a periodic process once the encrypted extract is ready and all sessions to the unencrypted extract are terminated.

When the state of an extract is changed to decrypt, a background job is created to decrypt the extract. The encrypted extract will allow user connections until the decrypted extract is ready and all sessions to the encrypted extract are terminated.

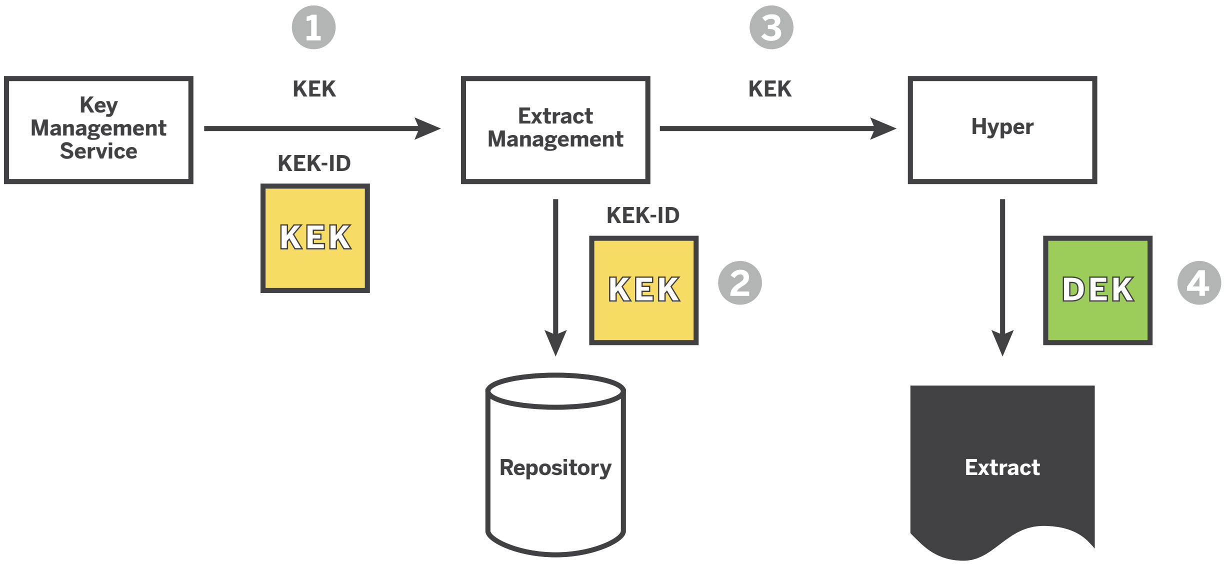
Encryption/decryption background jobs are created with normal priority on Server. The jobs may be preempted by other high priority jobs which may delay the encryption or decryption of an extract for an indeterminate amount of time, including extract refreshes.

When an encrypted extract is refreshed it will be encrypted on disk from the time it is generated. There is no extra step required.

When a request to download an encrypted extract is made from an authorized user, the extract is first decrypted on the Server, then the decrypted version is downloaded by the authorized user.

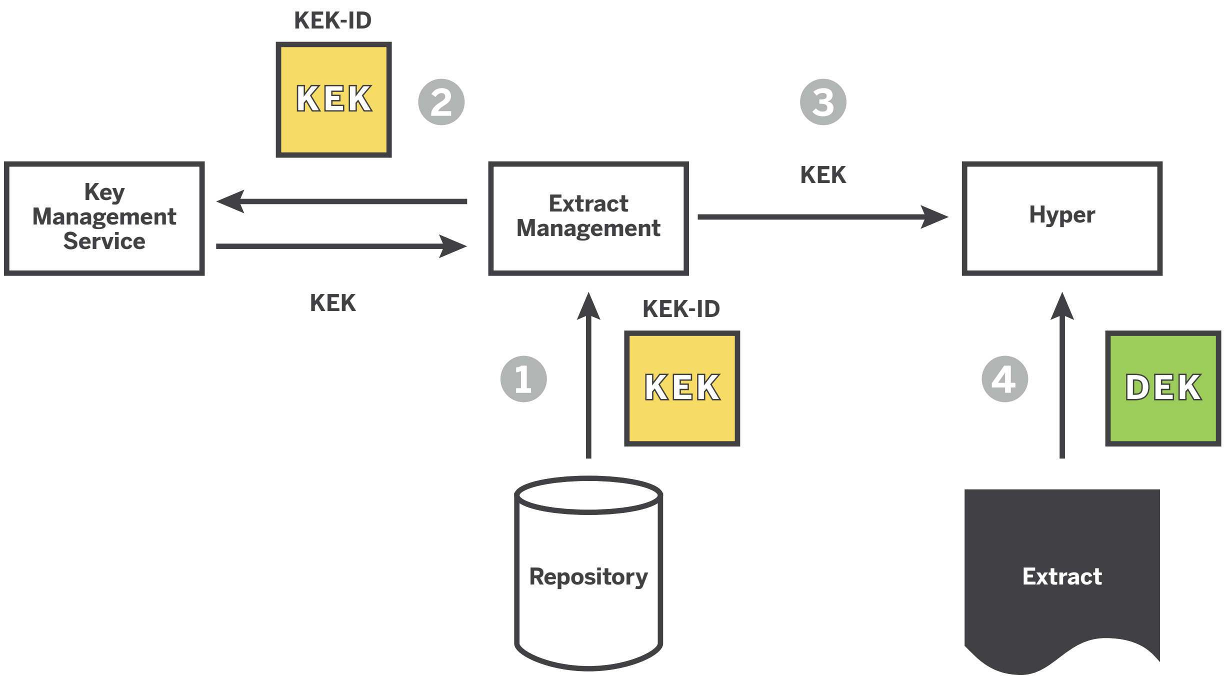
Security Considerations

Key Management



The Tableau Data Engine (Hyper) has been enhanced to perform encryption and decryption for .hyper files. When an encrypted extract is scheduled for creation, the extract management (EM) component requests a new Key Encrypting Key (KEK) from key management service (KMS). The KMS generates and returns the KEK to EM. In addition, the KMS encrypts the KEK using its own Master Extract Key (MEK) and returns the encrypted KEK to EM as the KEK-ID. EM stores the KEK-ID in the Repository along with the extract metadata. EM also tells the Data Engine to encrypt the extract with the provided KEK. The Data Engine generates data encryption keys (DEK) and uses them to encrypt the extract. Hyper encrypts the DEK with the KEK and stores the encrypted DEK with the extract. Each extract has a unique KEK, KEK_ID, and DEK. After the DEK and KEK are no longer in use each of the components forgets these keys.

Key Management return



When EM wants to open an extract, it retrieves the KEK-ID from the extract metadata stored in the repository. It then asks the KMS for the KEK associated with the KEK-ID. The KMS returns the KEK to EM which uses it in the call to open the extract in Hyper. The Data Engine retrieves the encrypted DEK from the extract file and decrypts it using the KEK. It then uses the DEK to decrypt the extract. Only data that is actually queried/used by the viz is decrypted and this is done on-the-fly. After the DEK and KEK are no longer needed each of the components forgets these keys.

There are two approaches to manage Master Keys (MK): Local KMS and AWS KMS. By default, Tableau Server uses Local KMS. If you have purchased the Server Management Add-On, you will have the option to use the AWS KMS implementation.

Local KMS

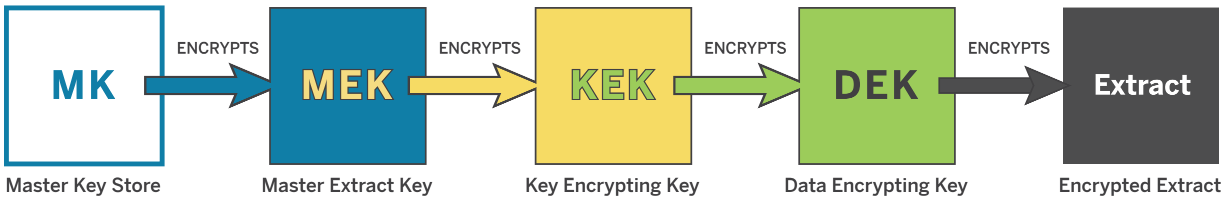


Figure 1 Local KMS Key Hierarchy

There is a chain of protection for the keys used to protect the extract that starts with the DEK generated by Hyper and ends with a Master Key (MK) that is managed by Tableau Server and stored in a key store on the filesystem (keystore.jks). The MK is used to encrypt one or more MEKs that are then stored in the configuration (tabsvc.yml). The most recent MEK is marked as the active MEK and is used to encrypt the KEKs into KEK-IDs. Other MEKs stored in the configuration are only used to decrypt KEK-IDs that were created in the past.

The local implementation stores its master keys on disk. All keys are encrypted at rest except for the Master Key (MK) which is the ultimate root of trust. The MK is stored on disk using file system access controls. Any operating system user with read permission to the MK file with access to the file system or TSM web interface can access keys needed to decrypt an extract. Users are not able to decrypt extracts if they do not have access to the TSM interface or the MK file.

It is recommended to set up user permissions such that maintainers of a system can access the Tableau Server file system but deny access to the file containing the master key. Low privileged maintainers of the system can see log files, files sizes of extracts and configuration data, but would not have access to the data within extracts.

AWS KMS

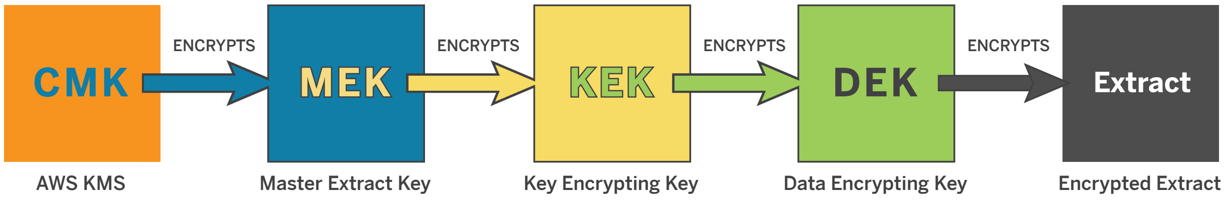


Figure 2 Key Hierarchy for AWS-KMS

For customers who host their Tableau Server in AWS, they will In the AWS KMS implementation, the Master Key (MK) is encrypted using a Customer Master Key (CMK) stored in AWS-KMS. We are looking to supporting other KMS providers in the future. The MEK can only be decrypted by a system that has been granted and IAM role that has the permission to use the appropriate KMS key to decrypt the MEK.

Encryption Algorithm and Module

Data encryption and key encryption uses the AES-GCM (NIST SP800-38D) FIPS 140-2 approved algorithm for encryption. Since this is an authenticating encryption algorithm it protects the confidentiality and integrity of the data. If the data is modified it will fail to decrypt. A 256-bit key is used with a 128-bit authentication tag. Extract encryption makes use of the OpenSSL crypto module and key management makes use of the Java Crypto Implementation. The crypto modules are not FIPS 140-2 validated.

Communication of Key

Outside of backup and administrative operations, the keys used for encryption never leave the Tableau Server cluster when using local KMS and will be decrypted using the AWS KMS service and sent over a secure connection in AWS KMS mode. Some of the communication between Tableau Server cluster nodes is currently unencrypted. In some cases, encryption keys (the KEK and MK) will be communicated over these unencrypted channels. Therefore, it is important that Tableau Server nodes be deployed with enough network security controls to prevent unauthorized parties from eavesdropping on or modifying those communications channels. It is recommended that Tableau Server nodes be deployed on their own secure subnet behind a firewall. Refer to the product documentation for more details.

Key rotation

Tableau Server provides a capability in the user interface that enables the administrator to re-encrypt all extracts on a site. Alternatively, administrators can use tabcmd to trigger key rotation for a site by issuing the following command:

```
tabcmd reencryptextracts <SITENAME>
```

When this process is triggered, background jobs are created to re-encrypt all encrypted extracts on the server. This will cause the generation of new KEKs and DEKs for each extract. The KEKs will be encrypted using the current MEK.

The current MEK can be rotated using the following tsm commands to manage server secrets:

```
tsm security regenerate-internal-tokens
```

This command will cause the current master key for the KMS (MEK) to be marked as read only and generate a new MEK to encrypt the KEKs. This command requires the Server to be stopped and restarted. The old MEK is kept around in read only mode to allow decryption of KEKs encrypted in the old MEK. This command will not trigger a rotation of all the KEKs and DEKs. Only new extracts and refreshes will be encrypted with the new MEK.

Backup and Restore

Extracts are stored in encrypted form in the backup image. Additionally, in order to maximize the availability of backups, all the information needed to restore a backup is included in the backup. This includes encrypted extracts key IDs (KEKS) and KMS data (MK and MEK) when using Local KMS. This means that someone with access to the backup file can access the encrypted extract data. Administrators must take care to secure their backups according to the organization's security policy. If your security policy requires you to rotate keys after restore, then refer to the product documentation <<INSERT LINK>> for instructions about how to re-encrypt extracts after a restore operation.

When using AWS-KMS, the master key (MK) is stored in AWS KMS and will not be present in the backup. Tableau Server must be configured with the appropriate information to access the required key in AWS KMS.

Export settings via the TSM interface does not export the key management data.

Site Import/Export

When a site is exported, the extracts are decrypted and exported in cleartext. Site export data must be handled securely for this reason. When the data is imported into the new site it will be encrypted according to the site's encryption policy. If the destination site is configured to 'Enabled', the default setting will depend on whether the extract was encrypted/unencrypted on the original site.

Providing Limited Access Accounts

When using local KMS, we recommend creating a lower privileged user that can access some files on the system including extracts and configs without being able to decrypt encrypted extracts or sensitive values in the config files. Lower privileged users should only be granted access to a subset of the configuration they need, making sure to exclude the local key store that holds the MK. For example, it is possible to give a user access to log files and other data files to check on disk resource usage without providing the ability to read the contents of the extracts. Using an external KMS (AWS KMS) also alleviates the need to do this.

Performance and Scaling Considerations

Encryption performance was tested on over 60K workbooks, which represent a mixed workload of short- and long-running queries, with most of the queries taking less than 1 second (98%). We compared workbook open times of encrypted extracts versus unencrypted extracts with the same data and workbook, for both opening the workbooks sequentially (once at a time) and ramping up the number of concurrent users on the same workbook.

The following is a summary of the key results we observed with further explanations below:

- The time to load a workbook the first time/for the first user changed marginally (3% for the .95 percentile). Our testing shows no performance impact for subsequent analysis (e.g. filtering) or more concurrent users hitting the workbook afterwards. Outliers might still exist.
- When encryption is first enabled or disabled for a site, there is an increase in processor and disk usage until the encryption process completes accounting for an additional load on Backgrounder nodes. Full and incremental refresh of extracts does not significantly impact performance.
- Encrypted extracts are not significantly larger than unencrypted extracts, however more temporary file space is required when this feature is enabled.
- Since encrypted extracts in backups cannot be compressed, the size of backups will increase roughly 2x.

The reason the overhead in the initial loading of a dashboard is only 3% in the .95 percentile is because of "streaming encryption". In other words, Tableau only decrypts the necessary data and does this only as needed when the data is fetched from disk. Right after fetching the first data block (a chunk of a column) from disk, it starts decrypting this block while waiting for the subsequent block to be fetched from disk. After that, the first block can be used to execute the query against. This decryption is much faster than the time it takes for data to be fetched from disk, hence, there is not much of a perceivable difference in terms of performance. The reason the overhead for additional concurrent users or subsequent analysis on that same dashboard does not impact performance is because the data is already decrypted in memory.

In our tests, the Backgrounder load during the encryption of all extracts measured about 50MB/s. This translates to a site with 100GB of extracts and only 1 Backgrounder needing about 30 minutes of spare backgrounder time. This work scales linearly with the number of Backgrounder processes (as long as Backgrounder processes are not CPU/disk bound), meaning that 2 Backgrounders (each having at least 1 CPU core for themselves) would complete the same encryption task in 15 mins, and so forth.

Additional security references

For additional security resources, including whitepapers, details on vulnerability reporting, and other documentation, visit: www.tableau.com/security