DATA ENCRYPTION

ENCRYPTION TECHNOLOGIES FOR
DATA PROTECTION ON THE NOW PLATFORM®
Executive summary
ServiceNow® is a software as a service (SaaS) company that provides robust data security and privacy capabilities to modern enterprises, transforming those businesses to run in the cloud. In today’s environment, there is no single encryption solution that addresses all data protection needs. Instead, the approach ServiceNow has taken is to provide customers with a suite of encryption options that can be used individually or in tandem to address a variety of data confidentiality use cases.

To meet the data security needs of modern enterprises—ranging from governmental regulatory and industry compliance objectives to implementing risk mitigation controls—ServiceNow offers encryption solutions at the application tier, database tier, and hardware tier.

At the application tier, your data is encrypted within the customer instance down to the database, so even an attacker with full software access to the database can’t read your data. Column-level encryption provides data encryption in our network. With ServiceNow Edge Encryption, your data is encrypted before it even reaches your ServiceNow instance in our network—this ensures there’s literally no way anyone at ServiceNow—or an attacker—can read your data.

At the database tier, ServiceNow Database Encryption encrypts your data directly in the database accessed by your applications running on your ServiceNow instance.

At the hardware tier, our Full Disk Encryption ensures data is encrypted at rest, thereby protecting you from a theft of storage attack.

Each approach has different pros and cons in terms of implementation, benefits, and functionality. This white paper will explain these solutions and provide the information you need to choose the correct ones based on your security needs.

Edge Encryption
Edge Encryption is a ServiceNow product that allows customers to encrypt columns (i.e., application fields) and attachments with encryption keys that customers own, control, and manage within their own networks outside of their ServiceNow instances.

Edge Encryption acts as a gateway between your browser and your SaaS ServiceNow instance. Traffic from your browser passes through the gateway on its way to the ServiceNow instance. The gateway, in turn, is configured to encrypt specific fields of data as they pass through. The traffic in the other direction is decrypted through the gateway, and users see plain text based on their request.

In more detail, Edge Encryption uses a proxy application hosted within your network. It encrypts data before the data is sent over the internet to your ServiceNow instance. The data remains in an encrypted state while in flight to the customer’s ServiceNow instance, which means the data continues to be encrypted at rest in the ServiceNow datacenter. The encrypted data is sent back, in its encrypted state, to the proxy application when it’s requested by a ServiceNow user and remains in the same encrypted state while it’s being sent to that user. Finally, the Edge Encryption proxy decrypts the encrypted data before sending it to the user within your network.

You can also tokenize data to mask a subset of particular data stored within a field. The tokenization process operates much like encryption—it’s performed by a proxy application that runs within the customer’s network and not in the ServiceNow datacenter where a customer’s ServiceNow instance resides.
Key features of Edge Encryption

- **Customer-owned**
  Customer-retained encryption key administration

- **Flexibility**
  Flexible encryption options to balance security and user operation requirements

- **Tokenization**
  Provides pattern-specific protection for structured data, such as credit card or Social Security numbers

- **API Support**
  REST and SOAP APIs to support additional system integrations, web services, and customizations

- **Easy administration**
  Easily administer and rotate encryption keys

- **Native platform**
  Tight integration within the ServiceNow platform architecture to support ServiceNow applications and the ServiceNow portal interface

- **Simple rule development**
  A native encryption rule development environment to provide integration support

Common use cases

- **Requirements that prohibit encryption keys from being stored in a cloud service provider**

- **Mitigating the risk of exposing sensitive data as either the result of a direct attack or of compromised data stored in a cloud**

- **Customers who need to comply with governmental and industry certification requirements and regulations**

- **Addressing the data sovereignty requirements for data that may be stored outside of a country’s domain**

**Perspectives on Edge Encryption**

When considering Edge Encryption, there are two important perspectives to consider using it from: your company’s infrastructure and the ServiceNow infrastructure.

**Your company’s infrastructure**

It’s imperative to examine the potential vulnerabilities of using Edge Encryption from the perspective of your company’s infrastructure. Although it’s not our purpose to provide an exhaustive list of prudent IT security practices here, the importance of protecting the user credentials used to connect to the ServiceNow instance through the Edge Encryption proxy cannot be overemphasized.
With Edge Encryption, your company’s encryption keys are applied for sessions that are connected through the Edge Encryption proxy. The proxy applies controls that secure the connections originating within the customer network. With this level of security, you can be sure your company’s data is securely encrypted and tokenized from within your own network infrastructure.

**The ServiceNow infrastructure**

The ServiceNow infrastructure provides additional protection when a user attempts to access sensitive data directly instead of connecting through the Edge Encryption proxy. Encrypted data that is at rest in the ServiceNow instance remains in an encrypted state and is shielded from exposure since the encryption keys do not reside on the instance. In this way, Edge Encryption provides data breach protection as well as reduces data sovereignty and compliance risks. So, if access to your company’s encrypted data were jeopardized, neither ServiceNow nor anyone else could examine the encrypted data and decrypt it without your encryption key—which is only accessible within your infrastructure.

Between the ServiceNow infrastructure and your company’s infrastructure, Edge Encryption addresses the big security issues companies face: data privacy, financial data protection, and data confidentiality. For example, a government agency that needs ServiceNow to decrypt its data may need to use Edge Encryption to keep the encryption keys to sensitive data within its own infrastructure. Figure 1 below depicts Edge Encryption in action from this perspective.
The Edge Encryption process

Figure 2 below shows the Edge Encryption process using a Social Security number field (SSN) that is configured with Edge Encryption within the ServiceNow customer instance. Whenever a ServiceNow user connected through the Edge Encryption proxy on your network enters or reads SSN data, the user sees the SSN in plain text because all encryption and decryption operations are processed by the Edge Encryption proxy. Upon closer examination, when the SSN field data is submitted to the ServiceNow instance, it is encrypted using an encryption key that resides on your premises.

As you can see on the left side of Figure 2, the data in the SSN field is converted from plain text to ciphertext. The data in the SSN field remains as ciphertext between the ServiceNow instance and where it is stored. When the user retrieves the SSN field, the data in the SSN field is processed through the Edge Encryption proxy using the same encryption key, then decrypted from ciphertext to plain text. Once the encryption is applied to the SSN field, ServiceNow can no longer see the SSN value as plain text, only its encrypted ciphertext. Because the encryption occurred prior to being entered into the ServiceNow instance, the instance only has a history of “000-00-0000” as “QUVTXzE2X2J.” The ServiceNow instance has no ability to decrypt this data because it has never had access to the required encryption keys.
Figures 3 and 4 show the user experience perspective when a field is encrypted with Edge Encryption. In this example, the short description field in the incident table was configured for encryption. When users are connected through the Edge Encryption proxy, they can see the content in plain text (Figure 3). However, if a user were to bypass the Edge Encryption proxy and access the same field directly, the short description would be presented as ciphertext (Figure 4). Even if the user bypassing the Edge Encryption proxy wanted to submit data in the short description field, the update would be rejected.

### Types of encryption

Edge Encryption provides three options that support the Advanced Encryption Standard (AES) for key lengths of 128 and 256 bits you can apply to data fields within an instance: standard, equality-preserving, and order-preserving encryption. All three use the customer-provided encryption key that resides on your premises, outside of the instance, and each includes its own capabilities and considerations.

- **Standard encryption** is the strongest of these encryption options as it produces random ciphertext for the value entered into a field. The tradeoff is that you’ll have a reduced ability to run any level of logic against the encrypted field. Another important consideration is that you can only encrypt attachments to records using the standard encryption option.

- The equality-preserving and order-preserving encryption options support logical operations and functionality on encrypted fields, but encrypted fields retain the same encrypted ciphertext value. Because of this, these two options provide less security than standard encryption.

- Equality-preserving encryption supports comparison operations, such as filtering, matching, and grouping. Order-preserving encryption does what equality-preserving encryption does but also supports sorting and less than/greater than operations. For a side-by-side comparison of these encryption options, see Appendix A.
Tokenization

Another layer of data protection that Edge Encryption provides is tokenization. During this process, Edge Encryption uses a randomly generated token to mask a particular predefined pattern of characters within a data field when the pattern is matched. The pattern itself is defined as a regular expression, and length-preserving random tokens are used to replace the tokenized value.

The token applied to the data contained in a field is stored on your premises in the same MySQL database your company uses for order-preserving encryption—never on ServiceNow instances. When tokenized data is stored on the instance, it will not appear in plain text while it’s in transit or at rest on ServiceNow instances.

Benefits of tokenization

Another benefit of tokenization is that it prevents your users from intentionally or unintentionally comingling sensitive data, such as Social Security numbers, within a ServiceNow instance. In addition, tokenization is an alternative option to apply where the field type is not supported by available encryption options.

The examples in Figures 5 and 6 illustrate tokenization from the user experience perspective. In this example, the patterns for a credit card and Social Security number were configured for tokenization. When the user connects through the Edge Encryption proxy, the content for those two values are displayed in plain text (Figure 5) but are actually tokenized in the instance. If the user were to bypass the Edge Encryption proxy and access the same incidents directly, the corresponding values within the short description field would be represented as a token (Figure 6).
Implementation considerations

While encrypting specific fields or tokenizing embedded strings of data is beneficial from a data security perspective, having ciphertext in place of actual data can lead to potential functional or operational challenges within the ServiceNow application. To avoid running into these challenges, follow the implementation considerations and suggested capability and configuration approaches provided in detail in Appendix B.

While the following list of known limitations is not intended to be exhaustive, we encourage you to do due diligence with respect to your own security and business requirements and corresponding impact when encrypting data.

Limitations

- Only string, journal, journal input, date, date/time, and URL fields as well as file attachments can be encrypted.
- Encryption occurs on a per-field basis only, and not as a group of fields within a record.
- Items that cannot be encrypted:
  - System tables, system fields in tables, choice fields, and virtual fields
  - Fields named “number” and fields associated with an automatic numbering scheme
- Encrypted data cannot be processed by back-end logic. If the database contains encrypted data, any business rule, back-end script, or back-end feature that relies on evaluating the data in the encrypted field will not run correctly.
- Encrypted fields cannot be changed by scripts run on the server.
- Encrypted data cannot be copied to a record where the field is not encrypted.
- Only Java KeyStore, SafeNet KeySecure, Unbound, and file store encryption key management are supported.
- An Edge Encryption proxy or defined group of Edge Encryption proxies may only support a single instance.
Column-level encryption

The ServiceNow subscription service includes a feature that allows users to encrypt columns (i.e., application fields) and attachments on their ServiceNow instances. With column-level encryption, the encryption key is stored and maintained within the ServiceNow instance.

These are the main features of column-level encryption:

- Supported field types (e.g., string, date, date/time, and URL) and file attachments can be encrypted using column-level encryption.
- It uses AES-128 or AES-256 encryption keys.
- You may supply your own column encryption keys, or the ServiceNow application will generate a random key.
- Access to encrypted data is based on the role assigned to the user.

Common use cases

Column-level encryption uses standard symmetric algorithms that encrypt columns, as well as any attachments within a ServiceNow application instance.

Encryption contexts

Column-level encryption introduces the concept of encryption contexts. An encryption context defines the key used by the encryption algorithm to encrypt the data in a field. You can use an encryption context to encrypt one or more fields and to encrypt attachments. You can also create more than one encryption context in a single ServiceNow instance, each of which may have a separate key.
Role-based encryption

Role-based encryption is a key benefit of column-level encryption. Figure 7 illustrates how role-based encryption is enabled.

Here are the results of these relationships:

- User 1 is a member of Role 1, which provides access to Encryption Context 1; this allows User 1 to see the contents of Field A and Field B.

- User 2 and User 3 are members of Group 1; Group 1 is a member of Role 1, which allows everyone in Group 1 access to Encryption Context 1 and allows User 2 and User 3 to see the contents of Field A and Field B.

- User 4 is not a member of any group or role and has no access to Encryption Context 1; not only does User 4 not have access to Field A or Field B, but User 4 will not even see that these fields exist.
The first example shows a simple configuration, but your use of the capability can be considerably more complex. Figure 8 shows multiple roles and multiple encryption contexts.

Here are the results of these relationships:

- User 1, User 2, and User 3 have access to Field A and Field B via Encryption Context 1.
- User 4 is a member of Role 1 and Role 2 and has access to all fields and to both encryption contexts.
- User 5 has access to Field C and Field D via Role 2.
Usage and restrictions

When you want to process sensitive data sets in the ServiceNow environment, you can use column-level encryption. The data is only decrypted when it’s viewed by a user with a role that is assigned the encryption context. This encryption is maintained throughout the backup process.

With column-level encryption, you can also apply the same encryption to attachments uploaded by users that you can apply to individual fields.

Controlling access to sensitive data often means limiting access either to a select group of users or granting it on an as-needed basis. Using role-based access control in conjunction with the column-level encryption feature ensures access to specific fields or attachments is only granted to the users who are assigned the appropriate roles.

Currently, ServiceNow does not support indexing or scheduled reporting data that is encrypted using column-level encryption. This is due to the nature of the data and ACL restrictions in the encrypted context; system accounts do not have access to encryption contexts.

Edge Encryption versus column-level encryption

This section serves as a guide to help you determine when to opt for column-level or Edge Encryption. At a high level, when enterprises want maximum control over the encryption of their data, Edge Encryption is the choice over column-level encryption. This is because the customer owns and controls the encryption key outside of the ServiceNow instance. However, depending on your requirements, using Edge Encryption may leave you with reduced functionality.

For example, column-level encryption can decrypt an encrypted column used in a server-side business rule when that rule is executed by a logged-in, interactive end user assigned the appropriate encryption context. However, Edge Encryption would not have this capability since the data needs to be decrypted on the instance to run the business rule. Table 1 shows a side-by-side comparison of the differences between Edge Encryption and column-level encryption functionality.

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Edge Encryption</th>
<th>Column-level Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption key controlled and owned by customer</td>
<td>YES</td>
<td>NO¹</td>
</tr>
<tr>
<td>Multiple levels of functional encryption for equality, filtering, grouping, and sorting operations</td>
<td>YES</td>
<td>NO²</td>
</tr>
<tr>
<td>Data tokenization based on defined encryption pattern</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Built-in encryption key rotation</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Encryption of standard out-of-the-box fields</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>REST/SOAP API encryption support</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Built-in mass encryption/decryption support</td>
<td>YES</td>
<td>YES³</td>
</tr>
<tr>
<td>Automatic attachment encryption</td>
<td>YES</td>
<td>NO⁴</td>
</tr>
<tr>
<td>Customer maintains additional infrastructure in their network to control encryption keys and encryption processing</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Decryption by server-side business rules</td>
<td>NO</td>
<td>YES⁵</td>
</tr>
<tr>
<td>Encryption/decryption based on user roles</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>

Table 1: Edge Encryption versus column-level encryption

¹Customer can define the encryption key
²Column-level encryption supports only equality filtering
³Where a single encryption context is used, mass encryption/mass decryption is supported for column-level encryption
⁴Manual process per record attachment for column-level encryption
⁵Supported only when business rules are executed by a logged-in, interactive end user assigned the appropriate encryption context
Database Encryption

Database Encryption encrypts all customer data at rest in the database with no impact to functionality. It utilizes the native capabilities of the database engine to encrypt data as it is written to the database and decrypt as it is read from the database using industry standard AES encryption. This technology, often called Tablespace Encryption or Transparent Data Encryption, is fully transparent to the customer and to the application. ServiceNow applications as well as custom applications can operate seamlessly without any changes necessary because the application always has access to the data it needs in the clear. When using Database Encryption all data is encrypted, including attachments, logs, and backups.

Customer Controlled Switch (CCS)

This additional cost option enables database encryption customers to supply their own symmetric database instance private key (DIPK). This may be desirable where customers have strict security requirements for key management, or to address obligations present in regulations like the US CLOUD Act.

The feature uses NIST best practices for safely transporting the DIPK, with a ServiceNow generated key-encryption key (KEK). The DIPK is wrapped with the KEK for secure transport. Customers using CCS are required to host a KEK-wrapped DIPK with a defined validity period on their own REST endpoint. A ServiceNow gateway securely retrieves the KEK-wrapped DIPK and generates a SHA-256 hash used to verify its integrity. The DIPK is held only in memory and never stored in a persistent, recoverable form. The ServiceNow gateway polls for the customer key every 15 minutes and if the key is inaccessible or expired, the instance is shut down automatically. Under these or any circumstances where the database is shut down, it remains in an encrypted state at rest.

Customers may also revoke the DIPK at any time and set the valid-until-date to whatever fits their unique policy requirements.

Using CCS can have very significant implications for instance availability in the case of inaccessible, expired, or lost customer keys. In the case of a lost key, ServiceNow has no means to recover the encrypted data whatsoever. These scenarios should be carefully considered by customers in their planning process before the implementation of CCS.
Common use cases

Encrypting all data at rest is a useful approach that provides a layer of security in cases where a large portion of the data in your environment is considered sensitive or when data may be considered sensitive in the future, such as due to regulations or changes in your business environment. Database Encryption is also useful in cases where it is critical to not impact functionality and application tier encryption is not necessary.

Database Encryption can be coupled with application tier encryption for a layered security approach. Highly sensitive fields that need to be encrypted at the application tier can be secured with Edge Encryption or column-level encryption. Layering encryption allows all data to be protected when not in use and highly sensitive fields, such as PII and PHI, to be protected from additional attack vectors.
**Full Disk Encryption**

Full disk encryption is provided via self-encrypting hard drives with AES-128 bit encryption. This delivers “at-rest” protection only and is focused on preventing data exposure through the loss or theft of hard disks holding customer data. It does not provide application tier protection for data in transit or against unauthorized access while the drive is operational. The key features of full disk encryption are:

- Encryption of the entire disk, which can only be decrypted by the operating system
- Does not impact the performance or functionality of the application

**Common Use Cases**

- Mitigate risk of sensitive data being exposed as a result of the physical theft of a disk drive used in a cloud instance

**Usage and restrictions**

Full-disk encryption is a high-speed encryption method integrated into ServiceNow’s Advanced High Availability (AHA) Architecture that provides encryption of customer data at rest. Full-disk encryption decrypts the data when actively being used or accessed by the server’s operating system. The hard drive models used by ServiceNow comply with the TCG Enterprise specifications and are secured using a passphrase generated from a key stored in our SafeNet key management appliance.

Full Disk Encryption can be coupled with both application tier encryption and database tier encryption for a layered security approach. Highly sensitive fields that need to be encrypted at the application tier can be secured with Edge Encryption or column-level encryption. Layering encryption allows all data to be protected when not in use and highly sensitive fields, such as PII and PHI, to be protected from additional attack vectors.
Summary

The use of data encryption solutions is an impactful decision and must be methodically thought through. With application tier encryption, selectively choosing which columns to encrypt and how to encrypt those can mitigate some of the issues discussed, but for others there may not be a mitigation strategy. Column-level encryption provides a user role-based approach to protecting data whereas with Edge Encryption data is also encrypted in transit and in use by applications running on the Now Platform. Database Encryption solves a different need at the database tier, making it possible to decrypt only the data required while leaving the remainder of your data not in use in an encrypted state. Full Disk Encryption solves the need to protect against physical theft of the hard drive, maintaining the data in a decrypted state while the hard drive is operational.

Appendix C provides a comparison of the ServiceNow data encryption solutions covered in this white paper. Collectively, these data encryption capabilities provide flexible and scalable security controls for protecting sensitive data running on ServiceNow solutions.
## APPENDIX A:

### Edge Encryption options

<table>
<thead>
<tr>
<th>Operations</th>
<th>Standard AES-128 or AES-256</th>
<th>Equality-preserving AES-128 or AES-256</th>
<th>Order-preserving AES-128 or AES-256</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group by</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Is empty</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Is not empty</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Equal</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Not equal (excludes empty fields)</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Is not</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Sort by</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is greater than</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is greater than or equal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is less than</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is less than or equal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contains</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Starts with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ends with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operators that imply the right side of the clause is a field</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Text search</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*MySQL is required for order-preserving encryption.*
## APPENDIX B:
### Functionality and encryption implications for Edge Encryption

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Implications</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reporting</td>
<td>Reporting operates on column data values. Because the ServiceNow application must use the column’s values to generate reports, there is the potential a report will not generate correctly because it does not have access to the clear text. This is only an issue if the report being generated uses columns that have been encrypted using Edge Encryption.</td>
<td>Review the columns you need to include in the report that may benefit from equality-preserving or order-preserving encryption, and use those supported functions where necessary. Do not export reports that contain encrypted columns since the report is generated on your instance without access to the encryption key.</td>
</tr>
<tr>
<td>Business rules and logic</td>
<td>ServiceNow runs all business logic on the back end, so any business rule that needs to read from or write to an encrypted column may have trouble executing the rule.</td>
<td>Review the columns included in business rules that may benefit from equality-preserving or order-preserving encryption, and use those supported functions where necessary. If this is not possible, do not use the encrypted columns.</td>
</tr>
<tr>
<td>Encrypted text exceeding table column widths</td>
<td>Encryption algorithms often create ciphertext that is longer than the plain text. For example, the name “King George III,” which is 15 bytes long, might be encrypted to “#]b</td>
<td>zg7s</td>
</tr>
<tr>
<td>Workflows</td>
<td>Similar to business rules, workflows often operate from a column’s value. A workflow that depends on the ability to examine plain text in a table column will fail to function because it only has access to encrypted versions of the text.</td>
<td>Review the columns from your workflows that may benefit from equality-preserving or order-preserving encryption, and use those supported functions where necessary. If this is not possible, do not use the encrypted columns.</td>
</tr>
<tr>
<td>Searching</td>
<td>ServiceNow executes all searches on the back-end database, which means all searches use the data within the columns. If the search is being executed against columns with ciphertext values rather than plain text values, a user may not receive the desired results. However, searches for exact matches will still work because the search term will be converted into ciphertext by Edge Encryption. This enables the back-end search function within ServiceNow to effectively search for the desired term. “Contains” searches on free-form text fields are the most difficult to implement because the search text cannot be found in the body of the encrypted text.</td>
<td>Tokenization can make “contains” searches possible. For example, a word or character string can be tokenized individually so the encrypted search text finds a matching tokenized word in the body of the field. Equality-preserving and order-preserving encryption provide a technique that partially addresses the “contains” search with strong encryption.</td>
</tr>
</tbody>
</table>
## APPENDIX B (continued):

### Functionality and encryption implications for Edge Encryption

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Implications</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorting</td>
<td>ServiceNow does all sorting on the back-end server. As an application, ServiceNow deals with large data sets and generally returns the Top N to the user based on some form of sorting. Because the application always sorts on the back end, and the application always sorts on the ciphertext values, when a user initiates the sorting of encrypted data, the results may appear incorrectly.</td>
<td>Apply order-preserving encryption to implement a technique that addresses this issue (while maintaining strong encryption) using a stored subset of plain text table data as a token to prepend to the ciphertext for sorting purposes before it is sent to the instance.</td>
</tr>
<tr>
<td>Bulk import/export</td>
<td>ServiceNow does all export and import activities on the back-end servers. As such, any exported data—Excel, XML, CSV, PDF, or other—exports the ciphertext values of any encrypted columns. Likewise, because these data formats are not supported, any attempt to import data into an encrypted column will result in unencrypted values being written into the column, unless the process that is sending data to the instance is configured to proxy communications through the Edge Encryption proxy.</td>
<td>Some vendor solutions are capable of intercepting exported data files, such as XML or CSV, and decrypting them prior to being delivered to the user. Check with your vendors to ensure they can encrypt and decrypt the file types you need. If they can, a web service integration is necessary.</td>
</tr>
<tr>
<td>Mobile access</td>
<td>To see any data that has been encrypted using Edge Encryption, a mobile browser must access the ServiceNow instance through the Edge Encryption proxy. Actions allowed via mobile devices need the ability to see the clear text data in order for the ServiceNow application to function correctly. This includes workflow approvals via mobile devices and other actions available to the user through the mobile interface.</td>
<td>Ensure that mobile access to the ServiceNow instance goes through the company’s network so all access is granted via the Edge Encryption proxy. Be selective about which columns you encrypt. Modify any workflows that use encrypted columns if the workflow is visible or accessible using mobile devices.</td>
</tr>
<tr>
<td>Inbound/outbound email and SMS notifications</td>
<td>When ServiceNow triggers a notification, it could send an email or SMS that contains a mixture of hard-coded plain text and encrypted field text. For example, an email template that looks like this: Dear $name, we have changed your shirt size from $old_size to $new_size. Will be rendered with field substitutions, so it looks like this if the corresponding columns are encrypted: Dear Bob Baker, we have changed your shirt size from 6^SD[&amp;%T to H7asdh78.</td>
<td>Edge encryption does not support inbound nor outbound email. Taking this into account, be selective about which columns you encrypt. Modify any SMS text message that uses encrypted columns and remove them from the message. Provide a URL in the message that leads to a ServiceNow page that shows the contents of the message—this way, the Edge Encryption Proxy can decrypt the text.</td>
</tr>
</tbody>
</table>
# APPENDIX B (continued):

## Functionality and encryption implications for Edge Encryption

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Implications</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference fields</td>
<td>Reference fields are not supported by Edge Encryption because the sysid that is being used to make the link between your form and the actual field needs to be in the clear.</td>
<td>Use a secondary field, encrypt it, and hide the reference from the form. The actual source field must be a string type and will need to be configured to be encrypted with one of the three available encryption types.</td>
</tr>
<tr>
<td>Web service integrations</td>
<td>ServiceNow can integrate with outside data sources using industry-standard web service protocols like REST and SOAP. A third-party integration, which is usually software running on a computer inside your network, can retrieve and insert data into ServiceNow automatically, but if that data is not properly encrypted, plain text can be inserted into columns that are expected to be encrypted. As a result, the Edge Encryption proxy attempts to decrypt text that was not encrypted in the first place. This leads to data inconsistencies within the ServiceNow instance and could impact what the user sees.</td>
<td>Configure all automated processes to send or receive data from the ServiceNow instance using encryption rules so the Edge Encryption proxy can identify the columns in the payload with the encrypted instances.</td>
</tr>
<tr>
<td>Legacy data</td>
<td>ServiceNow customers may have amassed large amounts of data within their ServiceNow instances within various columns. The amount of data these customers need to encrypt could contain millions of records. Because encryption keys and algorithms cannot be held within ServiceNow, encrypting large amounts of data using Edge Encryption can take a long time.</td>
<td>You can run a mass encryption job on a per-column and attachment basis. Plan when you want to run this type of operation carefully so you can accommodate for the volume of columns and attachments you plan to encrypt.</td>
</tr>
</tbody>
</table>
# APPENDIX C

## Data encryption solutions

<table>
<thead>
<tr>
<th>Description</th>
<th>Edge Encryption</th>
<th>Column-level encryption</th>
<th>Database encryption</th>
<th>Full disk encryption</th>
</tr>
</thead>
</table>
| **Type of data encrypted** | • String Text  
• Date  
• Date/Time  
• Attachments  
• URL  
• Journal | • String text  
• Date  
• Date/Time  
• Attachments  
• URL | All data is encrypted | All data is encrypted |
| **Supported key sizes** | AES-128 and AES-256 | AES-128 and AES-256 | AES-256 | AES-128 |
| **Tokenization** | Yes, for pattern-matched data | No | No | No |
| **Encryption key creation** | Customer | Managed by ServiceNow and the customer | ServiceNow | Self-encrypting drive |
| **Additional requirements** | • On-premises encryption proxy  
• Encryption key store  
• Optional on-premises MySQL Database for tokenization and order-preserving encryption | None | None | Dedicated environment |