Spark::red®

Oracle Commerce Hosting

PCI Compliance ATG Encryption Module

Version 1 for ATG 11.x

Date: May 28, 2015

Overview

The Payment Card Industry Data Security Standard (PCI DSS) is a set of requirements designed to enforce proper security measures that are taken by vendors to protect their customers' credit card data and prevent credit card fraud and identify theft.

All vendors who accept credit cards need to be compliant with all 12 groups of requirements in the PCI DSS and larger volume vendors will need to undergo PCI Compliance Audits from 3rd party security companies periodically. Failure to have an up-to-date compliance audit or failing an audit can result in fines or being unable to process credit cards, so ensuring your systems are PCI compliant is very important.

Cardholder Data Protection in Storage

The PCI DSS requirement group number three deals with protecting cardholder data in storage. This primarily covers requirements for storing credit card numbers securely.

Section 3.4 requires the credit card number to be encrypted using strong cryptography. Section 3.5 requires the encryption keys to be stored securely, encrypted themselves, and restrict access to the keys. Section 3.6 covers key management including periodic key changes.

The Spark::red PCI Compliance ATG Encryption Module (Crypto Module) covers all three of these requirements, utilizing AES 256-bit encryption, a data encryption passphrase and a key encryption passphrase used to encrypt the data encryption passphrase. The two passphrases are stored in separate systems on separate servers.

Batch Encryption Actions Supported by the Crypto Module:

- existing plain text data encryption;
- re-encryption of data which is currently encrypted with another non-PCI compliant system;
- re-encryption of data with a new passphrase based key (it makes the annual PCI mandated key rotation a simple matter).

ATG stores credit cards in two places by default: an order paid for with a credit card will store the credit card information with the order, and a user's profile can have one or more saved credit cards associated with it. Unfortunately by default ATG stores these credit cards in plain text, unencrypted. This is obviously not PCI compliant. ATG's new Commerce Reference Store (CRS) provides encryption for the user profile's stored credit cards, but not the order's payment credit card. The encryption is also far less secure than our module, and does not meet PCI requirements for key management and provides no ability for key rotation and re-encryption.

The Crypto Module allows you to easily make your ATG application PCI complaint and can encrypt your existing data, even that encrypted by other systems, easily.

Installation

Encryption, especially PCI compliant encryption, is difficult to get right. We have spent a great deal of time creating an easy to use module that provides strong encryption and supports PCI key management requirements. If you have any questions or concerns please email support@sparkred.com.

Since data worth encrypting is typically very important data, please ensure you have made backups before initiating any import or Rekey operations. Also, please be careful to not change or lose encryption passphrases. There are two passphrases, one in the .properties file for the Crypto Engine, and one stored in encrypted form in the database.

Please read the Technical Architecture section of this document to learn how the Crypto Module works and how key management is handled.

Installation Steps:

1. Install the Unlimited Crypto JCE Policy jars into your JRE.
2. Unzip the provided SR-Crypto.zip archive. It will expand into a directory named SPARKRED with a sub-directory named CRYPTO.
3. Uncompress the `jboss-bouncycastle-module.tgz` file in the SPARKRED/CRYPTO directory and move the resultant "modules/org/*" directory into your JBoss 6.1 EAP $JBOSS_HOME/modules/org/ directory.
4. Add a dependency on the module in your JBoss Deployment XML file (this may require custom ANT script work):

   ```
   <deployment><dependencies><module name="org.bouncycastle" slot="main" export="true"/>
   ```

5. Copy the SPARKRED directory and it's contents into your ATG installation's root directory.
6. Add SPARKRED.CRYPTO to your application's required modules list.

7. Run the SQL script located at SPARKRED/CRYPTO/sql/create-crypto.sql against your Core schema. This creates the table needed for encryption engine key management.

8. Build your application EAR and launch your application.

9. Please see the "Default ATG B2C Commerce Configuration" section of the documentation for additional information.

The Crypto Module can be installed in approximately 10 minutes.

Configuration

The Crypto Module provides encryption tools and infrastructure. Using this infrastructure to encrypt your data requires simple configuration.

Default ATG B2C Commerce Configuration

Since most people will want to start with encrypting the credit card data stored in the two default ATG credit card tables (DCSPP_CREDIT_CARD and DPS_CREDIT_CARD) we have included an additional module SPARKRED.CRYPTO.B2CCommerce which includes the configurations needed to encrypt the credit cards stored in those two table using a credit card instance of the encryption engine and utilizing a custom ATG Repository Property Descriptor to transparently encrypt and decrypt those two credit card number properties. The Crypto Module also includes overrides for the Profile and Order Repository definitions to use the new encryption property descriptor.

To take advantage of this default configuration you need make some additions to the basic installation steps above.

Additions to the Installation Steps:

1. Add SPARKRED.CRYPTO.B2CCommerce to your application's required modules list.

2. Run the SQL script located at SPARKRED/CRYPTO/B2CCommerce/sql/alter-oob.sql against your Core schema. This will increase the column size for the two credit card number columns in order to accommodate the encrypted values.

3. Configure the CreditCardCryptoEngine. You can see the default configuration here: SPARKRED/CRYPTO/B2CCommerce/config/sparkred/crypto/CreditCardCryptoEngine.properties. The values which need to be changed are the keyPassphrase: pick any passphrase you like, and the keyExpirationNotificationEmail: this is the e-mail of whoever should be notified when your key needs to be rotated within the next 30 days. We recommend that the keyPassphrase be managed securely, perhaps deployed on the production servers in a local ATG-Data config layer, and NOT checked into Subversion. In order to meet PCI requirements the key should be accessible by as few people as possible.

Custom Encryption Configuration

Create a custom Crypto Engine or a multiple Crypto Engines to provide encryption using different keys is easy. Please read the Technical Architecture section below to fully understand how the Crypto Engine works.

Configuration Steps:

1. Create a .properties file to configure your Crypto Engine ATG Component. Please use the ATG B2C Commerce Crypto Engine configuration as an example. This can be found in SPARKRED/CRYPTO/B2CCrommerce/config/sparkred/crypto/CreditCardCryptoEngine.properties. Your Crypto Engine configuration must have a unique cryptoEngineIdentifier and should use a unique keyPassphrase. The rest of the settings can be copied over, or altered as you see fit. You may wish to add this new component to your InitialServices list to start it automatically upon server start.

2. Start your ATG server. When the component is loaded, either from being in the InitialServices list, or being referenced from another component or the dynamo web admin, it will initialize itself, creating a secure data encryption passphrase and creating a record in the Crypto Repository.

You may now use your new Crypto Engine to encrypt and decrypt data securely, including credit card data.

Repository Property Encryption Configuration

While you can use a Crypto Engine component to programmatically encrypt and decrypt data, it's common to need to encrypt and decrypt data in the database accessed via ATG Repositories. To facilitate this the Crypto Module includes a CryptoPropertyDescriptor to allow easy transparent encryption and decryption for any String Repository property.

Once you have the Crypto Engine component configured that you wish to use for encrypting one or more Repository properties, you just need to alter or override the property definition in the Repository definition file. The new property-type is com.sparkred.crypto.CryptoPropertyDescriptor. It requires a new attribute called cryptoEngine which points to the Crypto Engine component you wish to use.

There is also an optional attribute called encryptOnly. If this is set to true, the data will only be encrypted when stored to the database and will NOT be decrypted when read from the Repository. This is for data you do not wish to show up in the ACC, BCC, dynamo web admin. This data will either be write only data you do not need to read again within your application, or data you will decrypt programmatically with your own code using the same Crypto Engine component the Repository property was pointing to.

Here is a stripped down version of Repository property definition for the creditCardNumber property of the OrderRepository which we use in our B2CCommerce module to handle the out-of-the-box ATG credit card properties:

```
<property

   column-name="credit_card_number"

   data-type="string"  name="creditCardNumber"

   property-type="com.sparkred.crypto.CryptoPropertyDescriptor">

     <attribute name="cryptoEngine" bean="/sparkred/crypto/
CreditCardCryptoEngine" />

     <attribute name="encryptOnly" value="false" />

</property>
```

Importing Data

Unless you are building a new ATG application, you probably already have data that you'd like to encrypt. The Crypto Module supports easy bulk importing of both plain text and encrypted data. You first need to configure a Crypto Engine as outlined above, or just use the default B2CCommerce CreditCardCryptoEngine.

Plain Text Data Importing

You can import and encrypt existing plain text data in your database using a few simple steps.

Plain Text Data Importing Steps:

1. If the tables are in use by your application you will need to bring down the site for a maintenance window. It is critical that no records are created or modified while the batch encryption process is underway.
2. Backup the tables with plain text data you are planning to encrypt, in case of a problem during the import/encrypt process.
3. Configure the RekeyEngine in the SPARKRED.CRYPTO.TOOLS module at /config/sparkred/crypto/tools/RekeyEngine.properties.
    - 3.1. The decryptorComponent property should be set to: decryptorComponent=/sparkred/crypto/tools/NoopCryptoEngine
    - 3.2. The decryptorMethod property should be set to: decryptorMethod=decrypt
    - 3.3. The engineToUpdate property needs to point to the Crypto Engine instance which will be responsible for the encryption of the data you are importing.
    - 3.4. The newKeyPassphrase should be set to the passphrase you configured for that Crypto Engine instance.
    - 3.5. The tableColumns should be configured for the data column(s) you need to encrypt. The property is a list and needs to include all columns which use

this particular Crypto Engine for encryption. The syntax is table_name.column_name.

3.6. The dataDataSource property should point to the DataSource which has access to the tables and columns listed in the tableColumns property. This is typically your Core schema and defaults to the standard ATG JTDataSource component.

4. Add the SPARKRED.CRYPTO.TOOLS to the modules list and start one ATG instance. Be sure your site does not direct customer traffic to this instance.

5. Go to the dynamo web admin page for the RekeyEngine: http://app-host:8080/dyn/admin/nucleus/sparkred/crypto/tools/RekeyEngine.

6. Be 100% sure all the previous steps were followed, all the tables listed in tableColumns were backed up, and you are ready to import.

7. Invoke the rekey method and wait for it to complete.

8. Shutdown the ATG instance.

9. Update your Repository definition or code to use the Crypto Engine to handle the data you encrypted.

10. Bring the site back up, and test.

11. If everything works correctly, delete the backup tables you made in the step 2.

Encrypted Data Importing

You can import and re-encrypt data which is already encrypted using another encryption engine. If you are using the ATG Commerce Reference Store (CRS) and the DESede encryptor it provides (/atg/store/security/DESedeEncryptor) please skip to the next section. If you are using another encryption system, please follow the subsequent steps:

Encrypted Data Importing Steps:

1. Bring down the site for a maintenance window. It is critical that no encrypted records are created or modified while the re-encrypt process is underway.

2. Backup the tables with encrypted data you are planning to encrypt, in case of a problem during the re-encrypt process.

3. Configure the RekeyEngine in the SPARKRED.CRYPTO.TOOLS module at /config/sparkred/crypto/tools/RekeyEngine.properties.

    3.1. The decryptorComponent and decryptorMethod properties need to point to the ATG Component and method which can decrypt the existing encrypted data. The decryptorMethod must take in a single String parameter (the encrypted data) and must return a String (the decrypted data). If your existing crypto system does not vend a method that matches that signature, you may need to write a wrapper component which will handle any configuration, key management, or data conversion needed.

    3.2. The engineToUpdate property needs to point to the Crypto Engine instance which will be responsible for the encryption of the data you are importing.

    3.3. The newKeyPassphrase should be set to the passphrase you configured for that Crypto Engine instance.

    3.4. The tableColumns should be configured for the data column(s) you need to encrypt. The property is a list and needs to include all columns which use this particular Crypto Engine for encryption. The syntax is table_name.column_name.

3.5. The dataDataSource property should point to the DataSource which has access to the tables and columns listed in the tableColumns property. This is typically your Core schema and defaults to the standard ATG JTDataSource component.

4. Add the SPARKRED.CRYPTO.TOOLS to the modules list and start one ATG instance. Be sure your site does not direct customer traffic to this instance.

5. Go to the dynamo web admin page for the RekeyEngine: http://app-host:8080/dyn/admin/nucleus/sparkred/crypto/tools/RekeyEngine.

6. Be 100% sure all the previous steps were followed, all the tables listed in tableColumns were backed up, and you are ready to re-encrypt.

7. Invoke the rekey method and wait for it to complete.

8. Shutdown the ATG instance.

9. Update your Repository definition or code to use the Crypto Engine to handle the data you encrypted.

10. Bring the site back up, and test.

11. If everything works correctly, delete the backup tables you made in step 2.

How To Re-Encrypt Data

In order to be PCI compliant, all encrypted credit card data must be "rekeyed" or re-encrypted with a new encryption key, at least every 12 months, if not more frequently.

The RekeyEngine in the SPARKRED.CRYPTO.TOOLS module allows you to perform easy batch re-encryptions of data in the database.

Re-Encryption Steps:

1. Bring down the site for a maintenance window. It is critical that no encrypted records are created or modified while the Rekey process is underway.

2. Backup the tables with encrypted data you are planning to Rekey, in case of a problem during the Rekey process.

3. Configure the RekeyEngine in the SPARKRED.CRYPTO.TOOLS module at /config/sparkred/crypto/tools/RekeyEngine.properties.

    3.1. The decryptorComponent and engineToUpdate both need to point to the Crypto Engine instance responsible for the encryption of the data you are Rekeying. If you have multiple Crypto Engines, you will need to repeat steps 3-11 for each Crypto Engine.

    3.2. The newKeyPassphrase should be set to a new passphrase.

    3.3. The tableColumns should be configured for the data column(s) you need to Rekey. The property is a list and needs to include all columns which use this particular Crypto Engine for encryption. The syntax is table_name.column_name.

    3.4. The dataDataSource property should point to the DataSource which has access to the tables and columns listed in the tableColumns property. This is typically your Core schema and defaults to the standard ATG JTDataSource component.

4. Add the SPARKRED.CRYPTO.TOOLS to the modules list and start one ATG instance. Be sure your site does not direct customer traffic to this instance.

5. Go to the dynamo web admin page for the RekeyEngine: http://app-host:8080/dyn/admin/nucleus/sparkred/crypto/tools/RekeyEngine.

6. Be 100% sure all the previous steps were followed, all the tables listed in tableColumns were backed up, and you are ready to rekey.

7. Invoke the rekey method and wait for it to complete.

8. Shutdown the ATG instance.

9. Update the configuration of the Crypto Engine whose data you rekeyed to have the keyPassphrase set to the newKeyPassphrase you configured in the RekeyEngine.

10. Bring the site back up, and test.

11. If everything works correctly, delete the backup tables you made in step 2.

Technical Architecture

The CryptoEngine component provides the core of the encryption module. Each CryptoEngine component is configured with a data encryption key, and can be utilized programmatically and/or through the CryptoRepositoryDescriptor to encrypt Repository properties.

When a CryptoEngine component starts, it looks for an existing Crypto Engine Configuration record in the CryptoRepository based on the cryptoEngineIdentifier property. If no record exists it initializes itself, generating a secure random data encryption passphrase, encrypting it using the key passphrase configured on the CryptoEngine, and storing it in the Crypto Repository. The data encryption passphrase is never displayed or known by anyone and is only stored in it's encrypted form. Both the key and data passphrases are not used as encryption keys. The passphrases are used to generate encryption keys using many passes of a secure hash algorithm.

About Spark::red

Spark::red is a managed hosting provider exclusively for the Oracle Commerce platform with every architect on the team possessing more than a decade of experience. Committed to industry best practices, we manage some of the highest revenue generating, most complex Oracle Commerce environments. Spark::red clients include multiple Fortune 1000 and IR Top 500 companies all over the world.

Spark::red's impressive growth and loyal client base speaks for itself. We are dedicated to providing exceptional Oracle Commerce hosting services and support backed by unmatched experience.

Sales | 888.666.5582 | sales@sparkred.com | www.sparkred.com

Headquarters

8201 164th Ave NE
Suite 200
Redmond, WA 98052
United States

Chelmsford

73 Princeton St
Suite 303
Chelmsford, MA 01863
United States

Boston

30 Newbury St
3rd Floor
Boston, MA 02116
United States