
GeoPackage Encryption Extensions

A. GeoPackage Encryption Extensions



The definition of the extensions are almost identical and therefore contain duplicated information. But the editor has chosen this documentation approach to make it clear that these extensions are actually independent from each other and that each will be packaged into a separate OGC document when being submitted to the GeoPackage SWG for starting the standardization effort.

The GeoPackage Encryption Extensions are the result from work within the OGC Disaster Pilot '21: The first extension defines SQLite schema rules for storing encrypted Features and the second extension for storing encrypted Tiles. Both extensions define how encrypted data (features or tiles) and the associated decryption key(s) as well as related metadata information can be stored.

The client implementation supporting the decryption of features and tiles was implemented by TODO. The GeoPackage Encryption Service (GES) is identified as deliverable D111-3 within the Disaster Pilot. The prototype implementation of this service (GES) is realized as a plugin for GeoServer, version 2.20. The plugin extends the OGC Web Map Service, version 1.1.0 interface to produce GeoPackage files including encrypted Tiles in PNG format and the OGC Web Feature Service, version 2.0.0 interface to produce GeoPackage files including encrypted Features with source encoding in GeoJSON.

The plugin was implemented by [Secure Dimensions](https://www.secure-dimensions.de)¹ and is available as Open Source from Github: [GeoServer Encrypted GeoPackage Plugin](https://github.com/securedimensions/geoserver-geopackage-encryption-plugin)².

An OpenAPI description of the GES was developed to illustrate the extended use of OGC WMS and WFS provided by Geoserver, v2.20. For the Disaster Pilot, a

¹ <https://www.secure-dimensions.de>

² <https://github.com/securedimensions/geoserver-geopackage-encryption-plugin>

simple demonstration that produces a GeoPackage with encrypted content for OSM places US is available for the [GES OpenAPI](#)³. The underlying GeoServer deployment is available [here](#)⁴.

A.1. GeoPackage Encryption Extension for Features



This extension is an OGC Disaster Pilot '21 result and may change radically.

A.1.1. Extension Title

GeoPackage Encryption Extension for Features

A.1.2. Introduction

This extension provides a storage scheme that allows to store Features, originally encoded in GeoJSON, as a SQLite BLOB. Each row in the features table(s) contains the encrypted Feature data and references to the metadata of the Data Encryption Key (DEK) via a foreign key to the `gpkg_ext_keys` table. The DEK data is encoded as a JSON Web Token ([RFC 7519](#)⁵) or JSON Web Encryption ([RFC 7516](#)⁶). As such, the JWT is digitally signed by the entity that created the key's metadata to ensure integrity and authenticity. The payload of the JWT contains information about the key and in particular a `kurl` that provides the link to fetch the DEK from the corresponding Key Management System (KMS). A JWE encoded information indicates that the actual DEK is encrypted in the JWE's payload using the JWK format.

A.1.3. Extension Author

Secure Dimension GmbH, in collaboration with the participants of the OGC Disaster Pilot 2021.

³ <https://ogc.secure-dimensions.com/geoserver-dp/api/>

⁴ <https://ogc.secure-dimensions.com/geoserver-dp/>

⁵ <https://datatracker.ietf.org/doc/html/rfc7519>

⁶ <https://datatracker.ietf.org/doc/html/rfc7516>

A.1.4. Extension Name or Template

`sd_encrypted_features` (will become `gpkg_encrypted_features` if adopted by the OGC)

A.1.5. Extension Type

New requirement dependent on [GeoPackage Core \(Clause 1\)](#)⁷ and [GeoPackage Schema Extension](#)⁸.

A.1.6. Applicability

This extension allows to store Simple Features, originally encoded in GeoJSON, as an encrypted BLOB. This extension does not constraint how the data is encrypted. It is outside the scope of this specification how the cipher key information is protected and securely communicated to the client application that is supposed to decrypt the data. Information about the encryption method (algorithm) and key length is available from the metadata of the DEK that can be obtained from the KMS in case a key reference is stored in the `gpkg_ext_keys` table or directly from the row in the `gpkg_ext_keys` table in case a JWE format is used.



For the OGC Disaster Pilot 2021, a working prototype was implemented that is described in more detail in the Disaster Pilot Summary ER.

A.1.7. Scope

read-write

A.1.8. Specification

The following sub-sections define how to store the encrypted content and the key information in a GeoPackage.

⁷ <http://www.geopackage.org/spec/#core>

⁸ http://www.geopackage.org/spec130/#extension_schema

Table *gpkg_extensions*

To use this extension, add the following rows to this table.

Table A.1. gpkg_extensions Table Rows

table_name	column_name	extension_name	definition	scope
gpkg_ext_keys	null	sd_encrypted_features	www.ogc.org/per/021-064.html#sd_encrypted_features	read-write
<user-defined encrypted features table>	null	sd_encrypted_features	www.ogc.org/per/021-064.html#sd_encrypted_features	read-write



The values in the definition column SHOULD refer in some human-readable way to this extension specification. If the extension is adopted by the OGC, it will gain the "gpkg_" prefix (replacing "sd_") and get a different definition permalink.

Table *gpkg_data_columns*

This table contains the information about the data type for the columns in the <user-defined encrypted features table> and gpkg_ext_keys table.

To use this extension, add the following rows to this table.

Row 1: Definition of the data column from the <user-defined encrypted features table> table

- table_name has the value <user-defined encrypted features table>
- column_name data
- name has the value <user-defined encrypted features table>-data
- title has value Encrypted Feature Data
- description description, e.g. The encrypted data of the feature

- `mime_type` has value `application/octet-stream`
- `constraint_name` has the value `null`

Row 2: Definition of the `data` column from the `gpkg_ext_keys` table

- `table_name` has the value `gpkg_ext_keys`
- `column_name` `data`
- `name` has the value `sd_encrypted_features-keys`
- `title` has value `DEK metadata`
- `description` **description, e.g.** The Data Encryption Key information represented as JWT or JWE
- `mime_type` has value `application/jose` to represent JWS, JWT or JWE encoding of the key's metadata
- `constraint_name` has the value `null`



No explicit media type exists for JWS or JWE encoded data. IANA media type "application/jose" must be used and from counting the dots, an application can defer whether the structure refers to JWS / JWT (2 dots) or JWE (4 dots). Some applications use the value "JWE" and "JWS" or "JWT" to determine between JOSE encodings. For this extension, the IANA media type "application/jose" is used.

New Table Definitions

Following are definitions of the tables for this extension.

As with other GeoPackage tables, this extension takes no position on how either of these tables are to be used by a client.

Table *<user-defined encrypted features table>*



The place holder `<user-defined encrypted features table>` refers to the name of the table that actually holds the encrypted feature information. This name varies but is identical for a GeoPackage instance.

This table contains encrypted features. The identifier is a string, uniquely representing the feature stored in this table. The identifier may be an obfuscation of the actual feature type in cases where naming the table after the feature type might disclose sensitive information.



For the Disaster Pilot '21 implementation, the table name corresponds to the `featureType` parameter of the WFS request and the identifier (`fid`) is the id of the actual feature that is encrypted.

The columns of this table are:

- `id` (type INTEGER) is the primary key
- `fid` (type TEXT) is the value of the original feature id
- `the_geom` (type BLOB) is the bounding box of the encrypted feature or null in case it is withheld for sensitivity reasons or not available
- `data` (type BLOB) is the encrypted feature
- `kid` (type TEXT) is the foreign key to the entry in the `gpkg_ext_keys` table

Table `gpkg_ext_keys`

This table contains the Data Encryption Key or its metadata. The primary key is used as foreign key by the `key_id` column from the `<user-defined encrypted features table>`.

The content of the `data` column contains either a JWT or a JWE. Typically, the format is either JWT containing metadata when managing the DEK at some other entity or a JWE when the DEK is encrypted inline.

The columns of this table are:

- `id` (type TEXT) is the primary key
- `data` (type TEXT) is the JWK representation either as JWT or JWE

A.2. GeoPackage DCS Tiles Extension



This extension is an OGC Disaster Pilot '21 result and may change radically.

A.2.1. Extension Title

GeoPackage Encryption Extension for Tiles

A.2.2. Introduction

This extension provides a storage scheme that allows to store Tiles, originally encoded in PNG, as a SQLite BLOB. Each row in the tiles table(s) contains the encrypted tiles data and references to the metadata of the Data Encryption Key (DEK) via a foreign key to the `gpkg_ext_keys` table. The DEK data is encoded as a JSON Web Token ([RFC 7519⁹](#)) or JSON Web Encryption ([RFC 7516¹⁰](#)). As such, the JWT is digitally signed by the entity that created the metadata to ensure integrity and authenticity. The payload of the JWT contains information about the key and in particular a `kurl` that provides the link to fetch the DEK from the corresponding Key Management System (KMS). A JWE encoded information indicates that the actual DEK is encrypted in the JWE's payload using the JWK format.

A.2.3. Extension Author

Secure Dimension GmbH, in collaboration with the participants of the OGC Disaster Pilot 2021.

A.2.4. Extension Name or Template

`sd_encrypted_tiles` (will become `gpkg_encrypted_tiles` if adopted by the OGC)

A.2.5. Extension Type

New requirement dependent on [GeoPackage Tiles Option \(Clause 2.2\)¹¹](#) and [GeoPackage Schema Extension¹²](#).

⁹ <https://datatracker.ietf.org/doc/html/rfc7519>


¹⁰ <https://datatracker.ietf.org/doc/html/rfc7516>

¹¹ <http://www.geopackage.org/spec130/#tiles>

¹² http://www.geopackage.org/spec130/#extension_schema

A.2.6. Applicability

This extension allows to store tiled data, originally encoded in PNG, as an encrypted BLOB. This extension does not constraint how the data is encrypted. It is outside the scope of this specification how the cipher key information is protected and securely communicated to the client application that is supposed to decrypt the data. Information about the encryption method (algorithm) and key length is available from the metadata of the DEK that can be obtained from the KMS in case a key reference is stored in the `gpkg_ext_keys` table or directly from the row in the `gpkg_ext_keys` table in case a JWE format is used.



For the OGC Disaster Pilot 2021, a working prototype was implemented that is described in more detail in the Disaster Pilot Summary ER.

A.2.7. Scope

read-write


A.2.8. Specification

Table *gpkg_extensions*

To use this extension, add the following rows to this table.

Table A.2. *gpkg_extensions* Table Rows

table_name	column_name	extension_name	definition	scope
gpkg_ext_keys	null	sd_encrypted_tiles	<code>https://www.ogc.org/per/021-064.html#sd_encrypted_tiles</code>	read-write
<user-defined encrypted tiles table>	null	sd_encrypted_tiles	<code>https://www.ogc.org/per/021-064.html#sd_encrypted_tiles</code>	read-write



The values in the `definition` column SHOULD refer in some human-readable way to this extension specification. If the extension is adopted by the OGC, it will gain the

"gpkg_" prefix (replacing "sd_") and get a different definition permalink.

Table *gpkg_data_columns*

This table contains the information about the data type for the columns in the `sd_encrypted_tiles-<user-defined encrypted tiles table>` table.

To use this extension, add the following rows to this table.

Row 1: Definition of the `data` column from the `<user-defined encrypted tiles table>` table

- `table_name` has the value `<user-defined encrypted tiles table>`
- `column_name` `data`
- `name` has the value `<user-defined encrypted tiles table>-data`
- `title` has value `Encrypted Tile Data`
- `description` **description, e.g.** `The encrypted data of the tile`
- `mime_type` has value `application/octet-stream`
- `constraint_name` has the value `null`

Row 2: Definition of the `data` column from the `gpkg_ext_keys` table

- `table_name` has the value `gpkg_ext_keys`
- `column_name` `data`
- `name` has the value `sd_encrypted_tiles-keys`
- `title` has value `DEK metadata`
- `description` **description, e.g.** `The Data Encryption Key information represented as JWT or JWE`
- `mime_type` has value `application/jose` to represent JWS, JWT or JWE encoding of the key's metadata
- `constraint_name` has the value `null`



No explicit media type exists for JWS or JWE encoded data. IANA media type "application/jose" must be used and

from counting the dots, an application can defer whether the structure refers to JWS / JWT (2 dots) or JWE (4 dots). Some applications use the value "JWE" and "JWS" or "JWT" to determine between JOSE encodings. For this extension, the IANA media type "application/jose" is used.

New Table Definitions

Following are definitions of the tables for this extension. As with other GeoPackage tables, this extension takes no position on how either of these tables are to be used by a client.

Table *<user-defined encrypted tiles table>*



The place holder *<user-defined encrypted tiles table>* refers to the name of the table that actually holds the encrypted tile. This name varies but is identical for a GeoPackage instance.

This table contains encrypted tiles. The identifier is a string, uniquely representing the tile layer stored in this table. The identifier may be an obfuscation of the actual tile layer in cases where naming the table after the tile layer might disclose sensitive information.



For the Disaster Pilot '21 implementation, the table name corresponds to the `layer` parameter of the WMS request.

The columns of this table are:

- `id` (type INTEGER) is a primary key
- `zoom_level` (type INTEGER) as specified by GeoPackage
- `tile_column` (type INTEGER) as specified by GeoPackage
- `tile_row` (type INTEGER) as specified by GeoPackage
- `data` (type BLOB) is the encrypted tile using symmetric key encryption with the key described in `dek_info` column.
- `kid` (type STRING) is the foreign key to the entry in the `gpkg_ext_keys` table

Table *gpkg_ext_keys*

This table contains the Data Encryption Key or its metadata. The primary key is used as foreign key by the `id` column from the `<user-defined encrypted tiles table>`.

The content of the `data` column contains either a JWT or a JWE. Typically, the format is either JWT containing metadata when managing the DEK at some other entity or a JWE when the DEK is encrypted inline.

The columns of this table are:

- `id` (type TEXT) is the primary key
- `data` (type TEXT) is the JWK representation either as JWT or JWE

