Procédure de déchiffrement des documents émis par Enedis sur les canaux numériques d'un Client ENTREPRISE

Identification : **Enedis-NOI-CF_107E**
Version : **1**
Nb. de pages : **6**

Version	Date d'application	Nature de la modification	Annule et remplace
1	01/06/2019	Création du document	-

Document(s) associé(s) et annexe(s) :

Résumé / Avertissement
Le chiffrement de données transportées via Internet permet de garantir la confidentialité des informations émises par Enedis vers un Client ENTREPRISE. Ces données sont par exemple contenues dans la pièce jointe d'un email, ou dans un fichier déposé sur un serveur FTP.

Ce document décrit :
- les modalités de chiffrement par Enedis des publications chiffrées vers les canaux numériques d'un Client ENTREPRISE
- les modalités d'obtention des clés utilisées lors de ces opérations de chiffrement/déchiffrement,
- les modalités de déchiffrement, par le Client ENTREPRISE, des fichiers chiffrés reçus.

Enedis - Tour Enedis
34 place des Corolles
92079 Paris La Défense Cedex
enedis.fr

SA à directoire et à conseil de surveillance
Capital de 270 037 000 euros
R.C.S. de Nanterre 444 608 442
Enedis est certifié ISO 14001 pour l'environnement

SOMMAIRE

1. Généralités

Enedis émet des flux (fichiers) chiffrés vers les Clients et Acteurs de Marché.
Les algorithmes de chiffrement utilisés par Enedis sont ceux reconnus en France par l'ANSSI (Agence nationale de la sécurité des systèmes d'information) comme assurant un haut niveau de confidentialité.
Pour le chiffrement vers les Clients ENTREPRISE, Enedis utilise l'algorithme AES (« *Advanced Encryption Standard* »), avec des clés de 256 bits.

2. Liste des Flux chiffrés vers les Clients ENTREPRISE

Le tableau suivant liste les flux chiffrés vers les Clients ENTREPRISE, et indique pour chacun :

- la méthode de chiffrement utilisée par Enedis ; cette méthode pouvant évoluer dans le temps, la date d'application est précisée dans la colonne « A partir du ». Les différentes méthodes de chiffrement sont décrites dans le chapitre 4 « Les différentes méthodes de chiffrement » ;
- la méthode de gestion des clés. Les différentes méthodes de chiffrement sont décrites dans le chapitre 3 « Les différentes méthodes de gestion des clés »

Code Flux	Nom Flux	A partir du	Chiffrement	Gestion des Clés
Enedis-FOR-CF_055E	Formulaire d'habilitation aux API Enedis	Juin 2019	7-Zip / Winzip	Clé par SMS
IFJ	Infra-J	30 Juil. 2019	AES256 IV dyn	Self-Care EC Entreprises
R171	Index quotidiens	Oct. 2019	AES256 IV dyn	Self-Care EC Entreprises
R172	Relevé de glissement	Oct. 2019	AES256 IV dyn	Self-Care EC Entreprises
R4Q, R4H, R4M	Publication Récurrente de Courbe de charge	Oct. 2019	AES256 IV dyn	Self-Care EC Entreprises

3. Les différentes méthodes de gestion des clés

A la date de création de ce document, deux méthodes de gestion des clés pour les Clients ENTREPRISE sont mises en place.

3.1. Méthode « Self-Care EC Entreprises »

Sur l'Espace Client ENTREPRISES, la page « Mes canaux de contact » permet à un Client ENTREPRISE de gérer, en *self-care*, ses canaux de contacts de type « Mail » et « FTP », à la maille d'un SIRET ou d'un SIREN.

Chaque canal de contact dispose de sa propre clé de chiffrement, qui est une clé symétrique de 256 bits qui doit rester secrète, connue seulement d'Enedis (pour chiffrer) et du Client (pour déchiffrer).

Cette clé est :

- créée lors de la création d'un nouveau canal de contact,
- modifiée lors de la modification d'un canal de contact existant : tous les flux reçus suite à cette opération seront chiffrés avec la nouvelle clé.

Pour la lire, il faut cliquer sur l'icône « Clé » du canal de contact : elle est alors affichée sous la forme de 64 caractères hexadécimaux, et un bouton « copier » permet de placer ces 64 caractères dans le « presse-papier » pour la coller dans le système d'information chargé du déchiffrement des publications reçus sur ce canal de contact. Il est important de noter que la clé n'est pas cette suite de 64 caractères, mais bien les 256 bits résultant de la conversion de ces caractères hexadécimaux en binaire (par exemple via la fonction Java `Hex.decodeHex()`).

La clé associée à un canal de contact sera utilisée pour chiffrer tous les flux chiffrés publiés sur ce canal de contact. Autrement dit, elle n'est pas associée à un « Code flux », mais au canal.

3.2. Méthode « Clé par SMS »

Cette méthode consiste à communiquer par SMS à un interlocuteur du Client ENTREPRISE la clé de déchiffrement de la pièce jointe chiffrée contenue dans un mail qui lui a été adressé en réponse à une demande ponctuelle.

Par exemple, lorsqu'un Client ENTREPRISE souhaite obtenir une habilitation à l'API Enedis « Infra-J » pour utiliser la prestation F375A ou P375A d'accès ponctuel aux données d'un compteur en infra-journalier, il renseigne le formulaire Enedis-FOR-CF_055E puis le transmet à Enedis, en indiquant dans ce formulaire :

- L'adresse mail de l'interlocuteur de l'ENTREPRISE chargé de recevoir la réponse d'Enedis (contenant les identifiants confidentiels)
- Le n° de mobile sur lequel ce même interlocuteur recevra le mot de passe de déchiffrement de la pièce jointe qui sera jointe au mail de réponse

Enedis créé l'habilitation et renseigne dans le même formulaire les identifiants confidentiels. Pour transmettre la réponse ainsi constituée à l'interlocuteur de l'ENTREPRISE, Enedis :

- Choisit une clé de chiffrement (suite de caractères non triviale) et utilise le logiciel libre « 7-Zip File Manager » pour créer une « archive » (au sens 7-Zip, à savoir un fichier d'extension «.zip ») contenant le formulaire chiffré avec cette clé
- Envoie cette archive par mail à l'interlocuteur de l'ENTREPRISE
- Envoie par SMS, à l'interlocuteur de l'ENTREPRISE, la clé de chiffrement, qui sera demandée par le logiciel « 7-Zip » ou « Winzip » pour déchiffrer l'archive.

4. Les différentes méthodes de chiffrement

Enedis utilise différents algorithmes de chiffrement.
Ce chapitre décrit, pour chaque algorithme, la méthode de chiffrement mise en œuvre par Enedis, la méthode préconisée de déchiffrement, et un exemple de code Java permettant d'implémenter le déchiffrement.
A la date de création de ce document, une seule méthode de chiffrement pour les Clients ENTREPRISES est mise en place.

4.1. Méthode « AES256 IV dyn » : chiffrement AES256 avec IV dynamique

4.1.1. Méthode de chiffrement

L'algorithme AES est utilisé en mode CBC avec une clé de 256 bits et un padding « PKCS5Padding ».
Pour chaque chiffrement d'un fichier, un IV (*Initialization Vector*) de 128 bits aléatoires est créé (chaque IV ne servira qu'une seule fois, on parle donc d'IV « dynamique »). La connaissance de l'IV est indispensable pour le déchiffrement, il doit donc être transmis au destinataire. Cette transmission se fait via le fichier chiffré transmis : les 128 bits de l'IV sont contenus dans les 128 premiers bits (ou 16 octets) du fichier transmis.

4.1.2. Méthode de déchiffrement

La méthode de déchiffrement consiste à utiliser l'algorithme AES256, en mode CBC (*Cipher Block Chaining*, ou Enchaînement des blocs), en utilisant l'IV lu dans les 128 premiers bits (ou 16 octets) du fichier chiffré reçu. Les données suivantes (après les 128 premiers bits) sont les blocs de données chiffrés en AES.

4.1.3. Exemple de code Java pour le déchiffrement

Le code Java suivant permet de déchiffrer un fichier chiffré en AES256 avec IV (*Initialization Vector*) de 128 bits en en-tête du fichier chiffré.
Il prend 3 paramètres en entrée :

1. Le chemin d'accès au fichier chiffré (reçu d'Enedis)
2. Le chemin d'accès au fichier déchiffré (qui sera créé)
3. La clé symétrique (de chiffrement/déchiffrement) de 256 bits

```
import java.nio.file.Files;
import java.nio.file.Paths;
import java.io.FileOutputStream;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;

/**
```

```
     * Dechiffre un stream en AES 256 avec la cle keyString
     *
     * @param encryptedStream = Fichier chiffré
     * @param clearStream = Fichier déchiffré
     * @param keyString = Clé AES256 encodée en Hexadécimal (64 caractères)
     * @throws CryptoException
     */

    public static void decryptStream(final InputStream encryptedStream, final
OutputStream clearStream,
                                     final String keyString) throws CryptoException {
        try {
            final byte[] keyBytes = Hex.decodeHex(keyString.toCharArray());
            final byte[] key = new byte[keyBytes.length];
            System.arraycopy(keyBytes, 0, key, 0, keyBytes.length);

            final SecretKey keyValue = new SecretKeySpec(key, "AES");

            final Cipher decryptCipher = Cipher.getInstance("AES/CBC/PKCS5Padding",
"SunJCE");
            // Lecture de l'IV depuis le 1er bloc du fichier d'entrée
            byte[] iv = new byte[AES256EncryptionUtilDyn.BLOCK_SIZE];
            encryptedStream.read(iv, 0, iv.length);
            IvParameterSpec ivspec = new IvParameterSpec(iv);

            decryptCipher.init(Cipher.DECRYPT_MODE, keyValue, ivspec);

            final byte[] buffer = new byte[1024];
            int noBytes = 0;
            final byte[] cipherBlock = new
byte[decryptCipher.getOutputSize(buffer.length)];
            while ((noBytes = encryptedStream.read(buffer)) != -1) {
                final int cipherBytes = decryptCipher.update(buffer, 0, noBytes,
cipherBlock);
                clearStream.write(cipherBlock, 0, cipherBytes);
            }

            final int cipherBytes = decryptCipher.doFinal(cipherBlock, 0);
            clearStream.write(cipherBlock, 0, cipherBytes);

        } catch (final Exception e) {
            throw new CryptoException("Erreur lors du déchiffrement des données", e);
        } finally {
            try {
                if (encryptedStream != null) {
                    encryptedStream.close();
                }
                if (clearStream != null) {
                    clearStream.flush();
                    clearStream.close();
                }
            } catch (final IOException localIOException) {
            }
        }
    }
```

4.2. Méthode 7-Zip / Winzip

4.2.1. Méthode de chiffrement

Cette méthode de chiffrement consiste à utiliser la fonctionnalité de chiffrement en mode AES256 offerte par le logiciel libre « 7-Zip » pour chiffrer un document à transmettre par mail au Client ENTREPRISE (à un interlocuteur désigné par lui), sous la forme d'une « archive » au sens « 7-Zip », qui se présente sous la forme d'un fichier d'extension « .zip ».

Attention, le mode AES256 intégré à 7-Zip n'est pas compatible avec le mode « AES 256 IV Dyn ». Il n'est donc pas possible d'utiliser 7-Zip / Winzip pour déchiffrer un document chiffré avec la méthode « AES256 IV dyn ».

La clé de chiffrement est choisie par Enedis au moment de l'envoi au Client ENTREPRISE et est communiquée par SMS à l'interlocuteur du Client ENTREPRISE, qui l'utilisera pour déchiffrer le document (la clé est en effet « symétrique »).

Ainsi, le document chiffré et la clé n'empruntent pas les mêmes canaux de communication.

4.2.2. Méthode de déchiffrement

Lorsque l'interlocuteur du Client ENTREPRISE procède à l'ouverture de l'archive, son logiciel de compression (7-Zip ou autre comme par exemple « WinZip ») reconnaît qu'il s'agit d'une archive chiffrée et demande le « mot de passe » : il faut alors saisir la clé reçue par SMS pour obtenir le document « en clair ».

Attention, le mode AES256 intégré à 7-Zip n'est pas compatible avec le mode « AES 256 IV Dyn ». Il n'est donc pas possible d'utiliser 7-Zip / Winzip pour déchiffrer un document chiffré avec la méthode « AES256 IV dyn ».