**AES**

La confidentialité des données chiffrées avec un algorithme symétrique repose uniquement sur la protection de la clé de chiffrement, si cette clé est découverte alors les données peuvent être déchiffrées. Cette clé permet de chiffrer et de déchiffrer les données. Avec AES - Rijndael la longueur de la clé peut être soit de 128, 196 ou 256 bits. Plus la longueur de cette clé est importante, plus il sera difficile de déchiffrer les données par "brut force". Pour une plus grande simplicité nous représentons une clé de chiffrement sous forme de texte, encodée en UTF-8 une clé :

* de 128 bits représentera 16 caractères
* de 196 bits représentera 24 caractères
* de 256 bits représentera 32 caractères

### Clé de chiffrement

Une clé de chiffrement de 128 bits pourrait être par exemple "abc123deaoezdf77". Cette clé doit rester secrète, et être aussi aléatoire que possible. Une clé de 256 bits sera beaucoup difficile à "casser" par attaque de type "brut force", cependant les performances seront moindres pour chiffrer des données avec une clé de 256 bits. Il faut donc trouver le bon équilibre entre robustesse de la clé et rapidité du chiffrement.

**Vecteur d'initialisation**

Un vecteur d'initialisation est un bloc de données aléatoires utilisé dans le processus de chiffrement du premier bloc de données. Cette séquence de bits, généralement de la même longueur que celle des blocs de données chiffrées, est souvent stockée sous forme de texte pour plus de facilité. Comme pour la clé de chiffrement, un vecteur d'initialisation de 128 bits représente 16 caractères UTF-8.

Le Framework .Net propose une implémentation de l'algorithme AES - Rijndael dans le namespace "System.Security.Cryptography". La classe implémentant cet algorithme ne nomme "RijndaelManaged". Voici les principales méthodes et propriétés de cette classe :

Méthodes et propriétés de la classe RijndaelManaged

* Méthode CreateEncryptor() : permet de créer un chiffreur AES.
* Méthode CreateDecryptor() : permet de créer un déchiffreur AES.
* Propriété Key : permet de définir ou d'obtenir la clé secrète, c'est elle qui permet le chiffrement et le déchiffrement.
* Propriété IV : permet de définir ou d'obtenir le vecteur d'initialisation. Un vecteur d'initialisation n'est pas utilisé systématiquement, il dépend du mode, le mode ECB n'en utilise pas.
* Propriété BlockSize : permet de définir ou d'obtenir la taille des blocs utilisés lors d'une opération de chiffrement ou de déchiffrement. Cette taille de bloc définit la taille maximale de données qui peuvent être chiffrées ou déchiffrées en une seule opération.
* Propriété Padding : permet de définir ou d'obtenir le mode de padding, c'est-à-dire le mode de remplissage des blocs quand un bloc n'est pas complet. Ces différents types de padding sont contenus dans une énumération. Voici la documentation MSDN des types de Padding :

|  |  |
| --- | --- |
| **Member Name** | **Description** |
| None | No padding is done. |
| PKCS7 | The PKCS #7 padding string consists of a sequence of bytes, each of which is equal to the total number of padding bytes added. |
| Zeros | The padding string consists of bytes set to zero. |
| ANSIX923 | The ANSIX923 padding string consists of a sequence of bytes filled with zeros before the length. |
| ISO10126 | The ISO10126 padding string consists of random data before the length. |

* Propriété Mode : permet de définir ou d'obtenir le mode de fonctionnement de l'algorithme, les modes disponibles (dans une énumération) sont les suivants : CBC; CFB; CTS; CBS; ECB et OFB. Voici la documentation MSDN des modes de chiffrement disponibles :

|  |  |  |
| --- | --- | --- |
| **Member Name** | **Description** |  |
| CBC | Le mode CBC (Cipher Block Chaining) introduit des commentaires. Avant que chaque bloc de texte brut ne soit chiffré, il est combiné au texte de chiffrement du bloc précédent à l'aide d'une opération de bits XOR. Cela permet de garantir que même si le texte brut contient de nombreux blocs identiques, chacun d'eux sera chiffré à l'aide d'un bloc de texte de chiffrement différent. Le vecteur d'initialisation est combiné au premier bloc de texte brut à l'aide d'une opération de bits XOR, avant que le bloc ne soit chiffré. Si un seul bit du bloc de texte de chiffrement est tronqué, le bloc de texte brut correspondant est également tronqué. En outre, un bit du bloc suivant, situé au même emplacement que le bit tronqué d'origine, est lui aussi tronqué. |  |
| ECB | Le mode ECB (Electronic Codebook) chiffre chaque bloc individuellement. En d'autres termes, tous les blocs de texte brut identiques qui se trouvent dans le même message, ou un message différent chiffré à l'aide de la même clé, sont transformés en blocs de texte de chiffrement identiques. Si le texte brut à chiffrer contient de nombreuses répétitions, le texte de chiffrement peut être déchiffré bloc par bloc. De même, tout pirate suffisamment tenace peut substituer des blocs individuels sans risque de détection. Si un seul bit du bloc de texte de chiffrement est tronqué, l'ensemble du bloc de texte brut correspondant est également tronqué. |  |
| OFB | Le mode OFB (Output Feedback) convertit de petits incréments de texte brut en texte de chiffrement, au lieu de convertir tout un bloc. Ce mode est similaire au mode CFB ; la seule différence entre ces deux modes réside dans la méthode de remplissage du registre à décalage. Si un bit du texte de chiffrement est tronqué, le bit de texte brut correspondant est également tronqué. Cependant, si des bits manquent dans le texte de chiffrement, ou si ce dernier contient des bits supplémentaires, le texte brut est tronqué à partir de cet emplacement. |  |
| CFB | Le mode CFB (Cipher Feedback) convertit de petits incréments de texte brut en texte de chiffrement, au lieu de convertir tout un bloc. Ce mode utilise un registre à décalage d'un bloc de longueur, qui se décompose en sections. Par exemple, si la taille du bloc est de huit octets, et qu'un seul octet est converti à la fois, le registre à décalage se décompose en huit sections. Si un bit du texte de chiffrement est tronqué, un bit du texte brut est également tronqué ; en outre, le registre à décalage est endommagé. Par conséquent, les incréments de texte brut suivants sont tronqués jusqu'à ce que le bit incorrect soit exclu du registre à décalage. |  |
| CTS | Le mode CTS (Cipher Text Stealing) gère toutes les longueurs de texte brut et produit un texte de chiffrement dont la longueur correspond à celle du texte brut. Ce mode a un comportement similaire au mode CBC, sauf en ce qui concerne les deux derniers blocs de texte brut. |  |

**Ceasar**