ENSTA Bretagne
2, rue François Verny
29806 BREST cedex
FRANCE
Tel +33 (0)2 98 34 88 00
www.ensta-bretagne.fr

UV4.8

promo 2017

8 mai 2016

AES-GCM

RIGAUD MICHAËL et BADIER CHARLIE

Table des matières

Introduction

GCM ou Galois Counter Mode est un mode d'opération de chiffrement par bloc en cryptographie symétrique. C'est un algorithme de chiffrement authentifié qui garanti l'intégrité et l'authenticité des données. Lors des opérations que nous verrons plus loin, cet algorithme demande de chiffrer avec un autre algorithme de chiffrement. D'après la norme IEEE 802.1AE, on utilise l'algorithme AES (Advanced Encryption Standard). On appelle donc cet algorithme AES-GCM .

Dans ce rapport, nous expliquerons dans un premier temps le fonctionnement de AES-GCM ainsi que de ses autres modes. Puis nous le comparerons à d'autres algorithmes semblables en termes de complexité. Enfin, nous essayerons de voir quels sont les principaux vecteurs d'attaques de cet algorithme dans les applications usuelles.

CHAPITRE 1

Fonctionnement

L'algorithme AES-GCM possède deux modes, le premier est le mode courant GCM et le second est le GMAC.

GCM

Avantanges de GCM

AES-GCM est un algorithme qui assure un haut niveau de sécurité grâce à AES, mais surtout il assure l'authenticité et l'intégrité des données. C'est à dire que si Alice essaye de communiquer avec Bob, elle est assurée que Charlie ne pourra pas lire ses données mais également qu'il ne pourra pas les modifier sans que Bob s'en aperçoive.

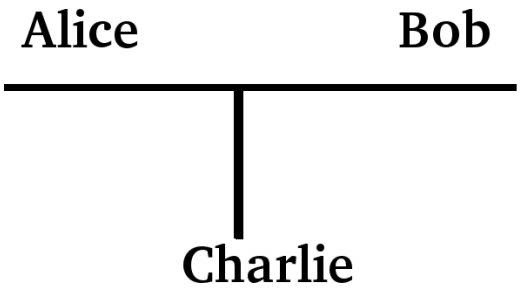

FIGURE 1.1 – Bob et Alice

De plus, GCM est un algorithme parallélisable qui assure une implémentation à haut débit à la fois matériel et logiciel.

Néanmoins, AES-GCM suppose que Alice et Bob se soient au préalable échangés une clef secrète.

Acronymes

Tout d'abord, pour bien expliquer le fonctionnement de AES-GCM , il nous faut définir certains acronymes.

Plaintext	le texte à chiffrer
Ciphertext	le texte chiffré
auth Data	des données supplémentaires à authentifier
K	la clé de chiffrement (secret)
H	Sous clé de hachage (secret)
IV	Vecteur d'initialisation supposé aléaoire
Mult	une multiplication dans l'espace de Galois

Chiffrement

AES-GCM est composé de deux blocs distincts, le bloc de chiffrement et le bloc d'authentification [1]-intégrité [2].

Dans un premier temps on va parler du bloc de chiffrement. Dans GCM il y C pour « counter », c'est-à-dire que AES-GCM s'appuie sur un mode qu'on nomme CTR [3].

Ccomme on peut le voir sur la figure 1.2, ce mode chiffre un compteur avec une clef (K) à travers un algorithme. Ensuite il réalise une opération de type XOR (ou exclusif) sur le texte clair avec la sortie de l'algorithme pour obtenir le texte chiffré.

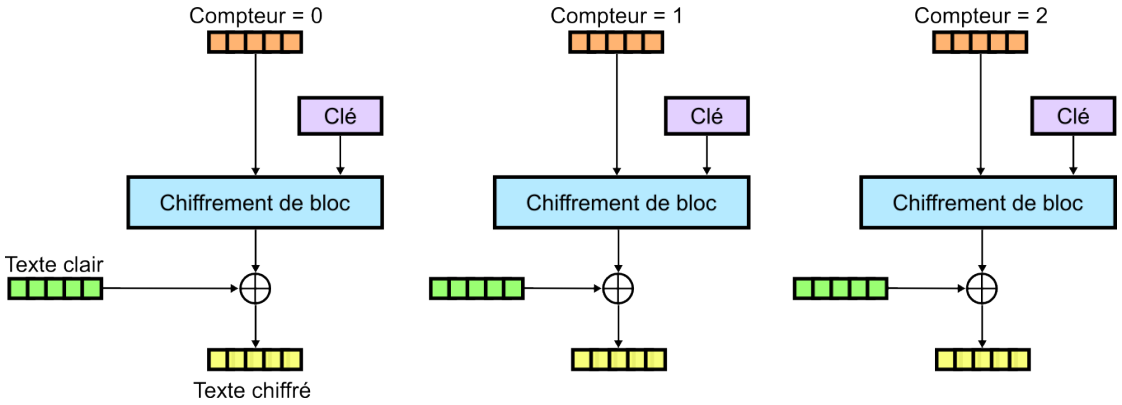

FIGURE 1.2 – schéma CTR [**?**]

Ce mode combine les avantages du chiffrement par flots, est pré-calculable et est parallélisable. En effet il est possible de calculer à l'avance en parallèle tous les chiffrés des compteurs. Il ne restera plus qu'à les passer dans la fonction XOR avec le clair pour obtenir le chiffré.

Dans le cas de AES-GCM le bloc de chiffrement est l'algorithme AES, et le compteur est le vecteur d'initialisation pseudo aléatoire IV qu'on incrémente.

Authentification-Intégrité

L'algorithme GCM permet de créer un Tag qui valide l'authenticité et l'intégrité des données. Pour mieux comprendre nous avons découpé cette opération pour voir tous les éléments qui interviennent dans ce processus.

Tout d'abord, la première chose dont on cherche à s'assurer est l'intégrité des données chiffrées que nous envoyons. En effet, nous avons chiffré notre texte mais rien ne nous protège contre une modification intentionnelle ou accidentelle du message envoyé. Ainsi, si Charlie cherche à gêner la communication de Bob et Alice et qu'il change certains bits on voudrait s'en rendre compte. On pourrait réaliser un hash de notre message avec des

1. On parlera ici d'authenticité lorsque l'on veut s'assurer que le message vient de la bonne personne
2. On parlera d'intégrité lorsque l'on veut s'assurer que les données n'ont pas été modifiées
3. CounTeR

fonctions comme sha ou md5, mais si on fait cela on ne pourra pas être protégé contre les modifications intentionnelles. Charlie n'aurait qu'à remplacer le hash par le hash du message contenant sa modification. La solution retenue dans GCM est d'utiliser des multiplications dans l'espace de Galois avec une autre clef nommée H secrète. On fait donc passer le premier bloc de chiffrés dans un bloc de multiplication avec H. Puis on réalise un XOR du résultat avec le bloc de chiffré suivant. Enfin on refait la multiplication comme on peut le voir sur l'image 1.3.

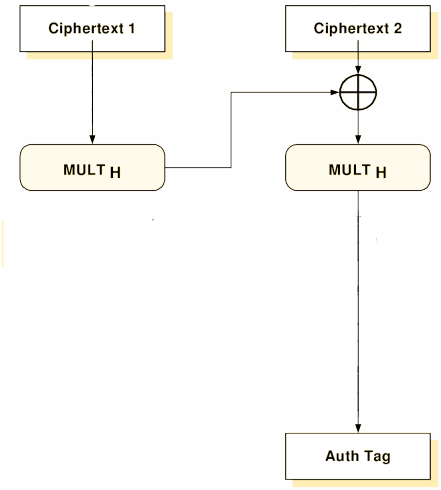

FIGURE 1.3 – authentification du message chiffré

Mais on ne veut pas seulement vérifier l'intégrité du message mais également du vecteur d'initialisation IV. Pour cela on réalise un « et » logique entre la longueur de message et la longueur de l'IV puis un XOR avec le tag précédent. Enfin, on effectue un bloc de Multiplication dans l'espace de Galois toujours avec la clef H comme on peut le voir sur l'image 1.4.

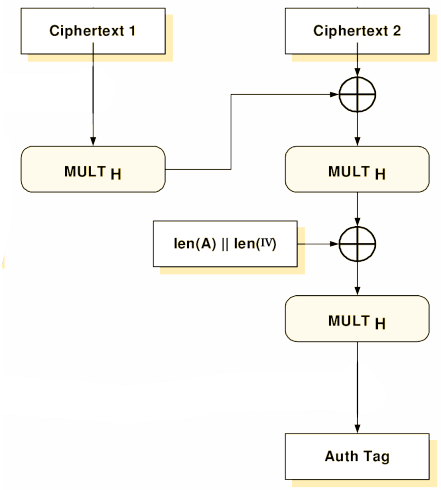

FIGURE 1.4 – authentification de la longueur du message et de IV

Ensuite, lors d'une communication entre Alice et Bob il y a des protocoles qui sont utilisés pour communiquer comme TCP/IP. Il y a donc des données qui vont entourer le message comme l'adresse IP qui permettent d'authentifier l'émetteur. Pour être certain que le message n'a pas été intercepté, on va intégrer ces données à notre tag d'intégrité et ainsi assurer l'authenticité du message. Au début de l'algorithme, on fait passer ces données dans un bloc de multiplication puis on réalise un XOR avec le premier bloc chiffré comme on peut le voir sur l'image 1.5.

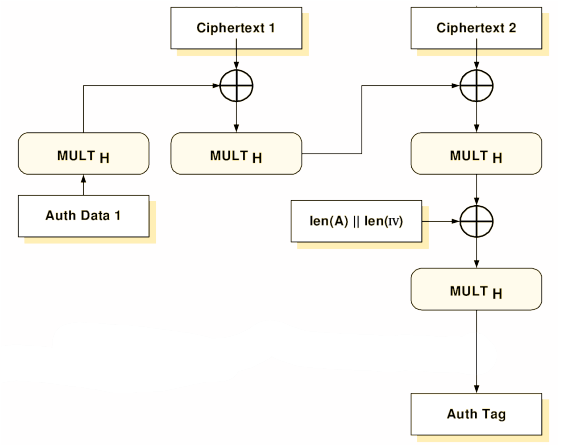

FIGURE 1.5 – authentification de data supplémentaire

Nous avons donc l'intégrité et l'authenticité du message, de la longueur de IV, de la longueur du message, et des données périphériques au message, mais pour terminer l'intégrité des données on va ajouter à ceci l'IV et la clef K. Pour cela nous chiffrons l'IV avec un compteur à 0 avec la clef K à travers l'algorithme AES. Puis nous réalisons un XOR avec le tag précédent comme on peut le voir sur l'image 1.6.

On obtient donc un tag qui permet de vérifier l'intégrité de tous les paramètres du message et l'authenticité de l'émetteur.

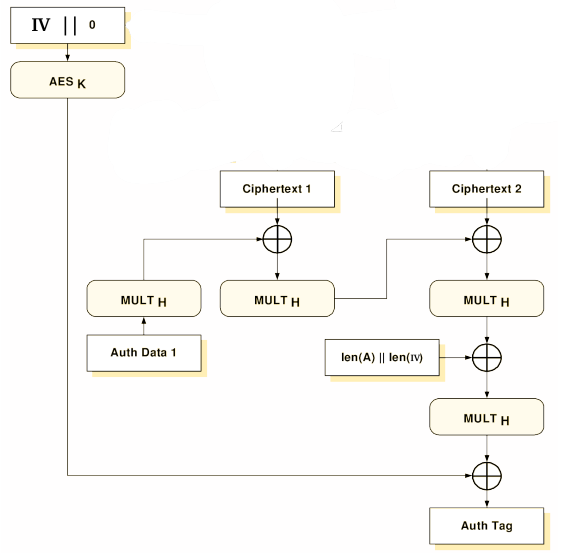

FIGURE 1.6 – authentification de la clef K

Le fonctionnement global de AES-GCM est résumé sur l'image 1.7.

Déchiffrement

A COMPLETER

GMAC

GMAC est un cas particulier de GCM où aucun texte brut n'est présenté.

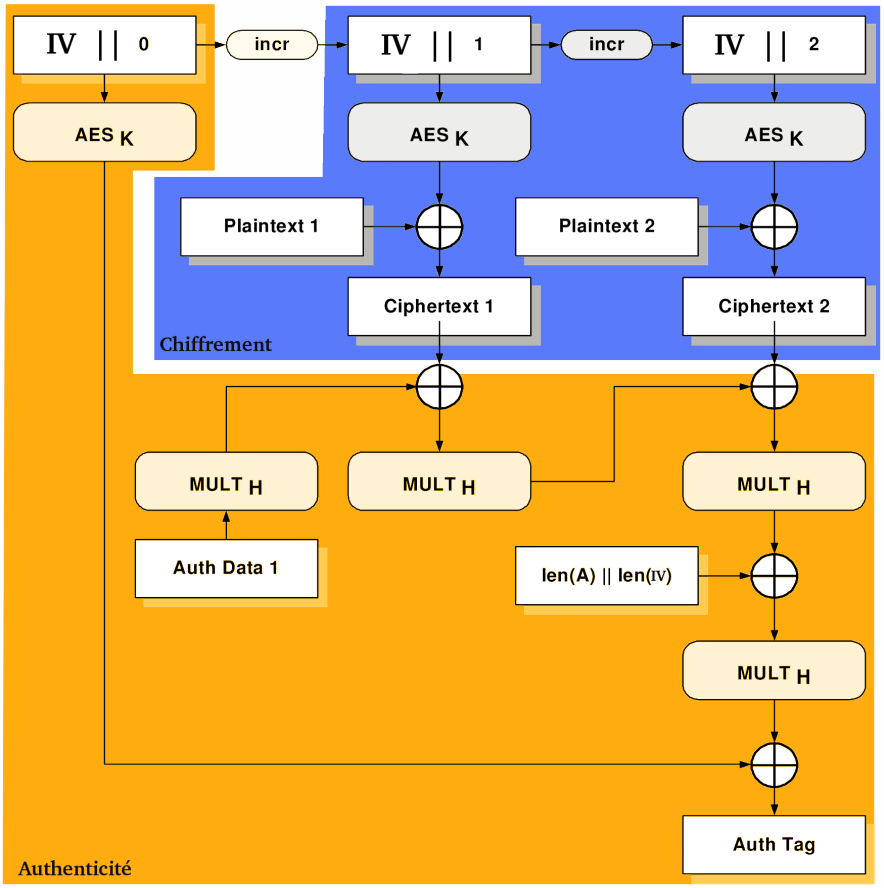

FIGURE 1.7 – Fonctionnement de GCM

CHAPITRE 2

Les différents modes

Il existe plusieurs modes de fonctionnenement de l'AES, comme l'AES-GCM .

ECB - Electronic codebook

Le mode ECB (Electronic codebook ou dictionnaire de code) est le plus simple. Il consiste à diviser le message à chiffrer en blocs qui vont être chiffrés indépendament les uns des autres. Pour le déchiffrement on procédera de la même manière en découpant le texte chiffré en blocs et en décryptant les blocs indépendament les uns des autres.

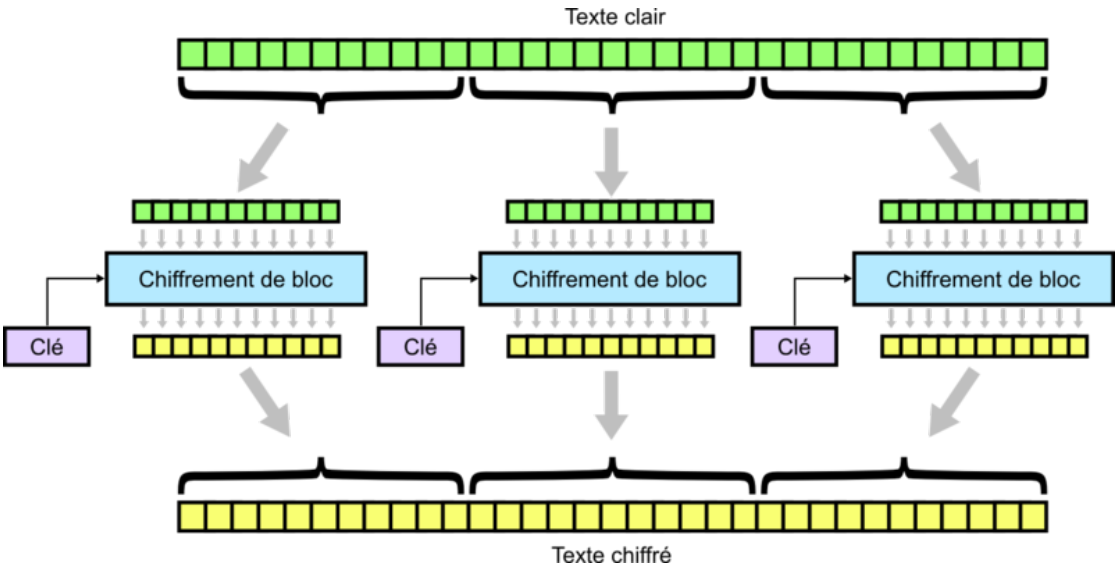

FIGURE 2.1 – schéma ECB [**?**]

Ce mode présente possède les avantages du chiffrement par flots, est pré-calculable et est parallélisable. Il offre la possibilité de déchiffrer une zone quelconque du texte chiffré et ainsi de déchiffrer une partie seulement des données.

Cependant ce mode possède un défaut considérable : deux blocs de texte clair seront chiffrés de la même manière, car il n'y a pas de randomisation. Ce défaut rend le mode ECB vulnérable aux attaques par dictionnaire et à l'analyse fréquentielle. En effet pour une clef donnée, on pourra générer un dictionnaire avec les correspondances entre les clairs et le chiffrés, permettant ainsi de retrouver le texte clair. Pour ces raisons l'utilisation de ce mode est fortement déconseillé.

CBC

Avec le mode CBC (Cipher Block Chainning ou Enchaînement des blocs), on applique à chaque bloc de texte clair un "XOR" (ou exclusif) avec le bloc chiffré précedent. Ainsi chaque bloc chiffré dépend des blocs traités auparavavant. Pour le premier bloc il faut fournir un vecteur d'initialisation.

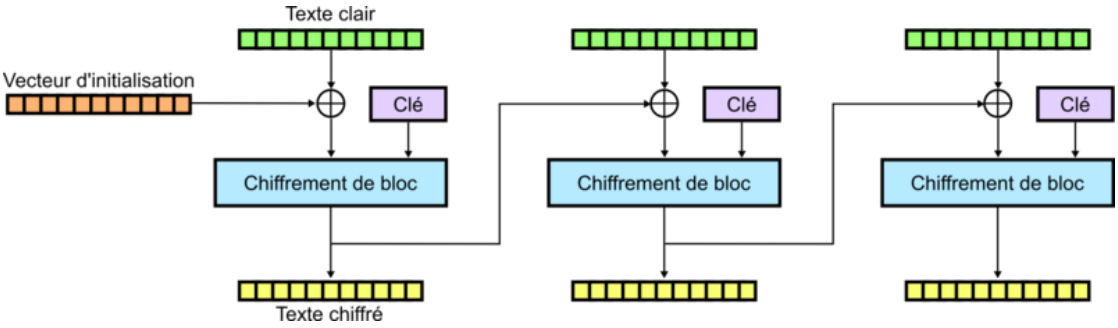

FIGURE 2.2 – schema CBC - Chiffrement [**?**]

Ce mode présente possède les avantages du chiffrement par flots, et il offre également la possibilité de déchiffrer une zone quelconque du texte chiffré. Cependant un des inconvénients est que le chiffrement est séquentiel (c'est-à-dire il ne peut pas être parallélisé).

Pour le déchiffrement, on passe le premier bloc crypté dans le déchiffrement de bloc et on effectue un "XOR" avec le vecteur d'initialisation IV. Dans le cas où le vecteur d'initialisation est incorrect seul le premier bloc crypté sera impossible à décrypter. En effet à chaque bloc on applique un "XOR" avec le chiffré du bloc précédent, et pas le texte clair. Ainsi on peut retrouver un bloc de texte clair uniquement à partir du bloc crypté précédent, ce qui permet ainsi la parallélisation de la décryption.

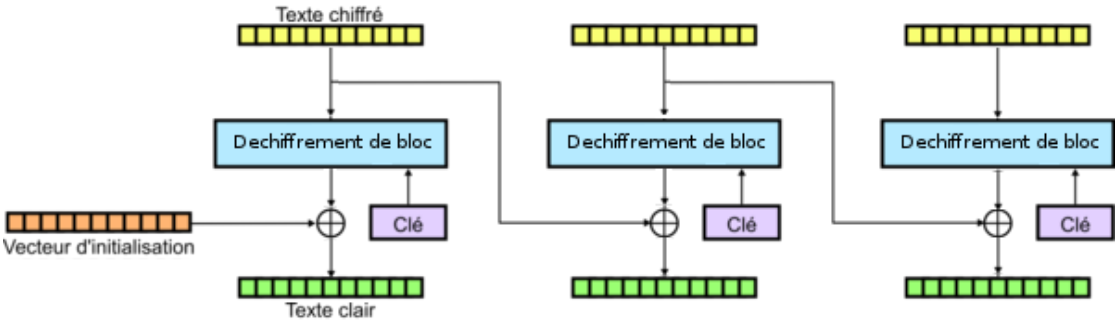

FIGURE 2.3 – schema CBC - Déchiffrement

CFB

Le mode CFB (Cipher FeedBack ou Chiffrement à rétroaction) est similaire au mode CBC. Tout comme le CBC, ce mode permet de déchiffrer n'importe quelle zone du chiffré. Cependant, comme le CBC, le chiffrement est séquentiel, il ne peut donc pas être parallèlisé. Le déchiffrement est similaire au CBC et peut, quant à lui, être parallélisé.

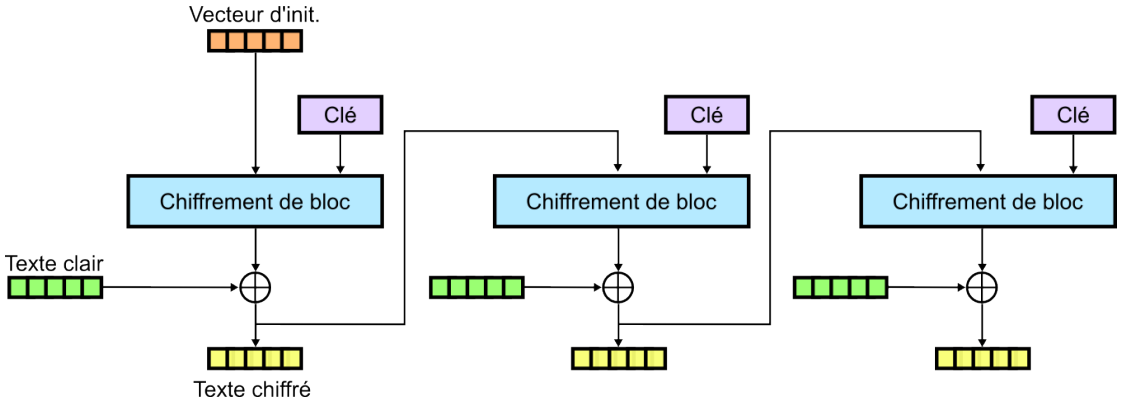

FIGURE 2.4 – schema CFB - Chiffrement

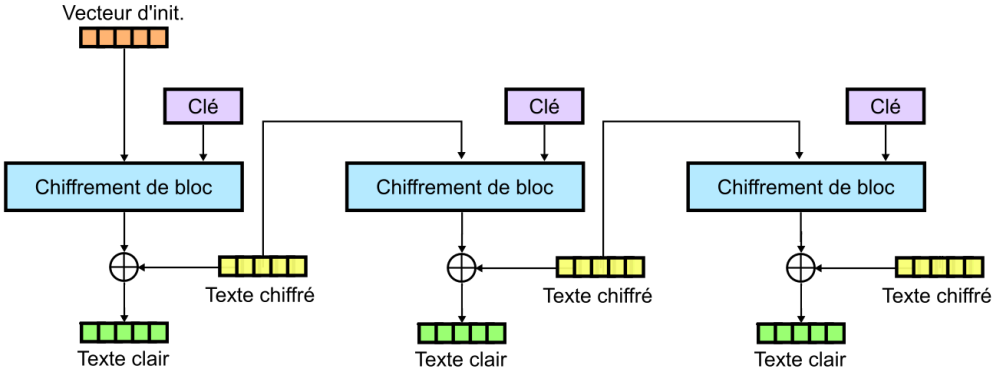

FIGURE 2.5 – schema CFB - Déchiffrement

OFB

Le mode OFB (Output FeedBack) est une variante du mode CFB. En effet, au lieu d'utiliser un bloc chiffré pour chiffrer le suivant, le mode OFB va utiliser le chiffré du vecteur d'initialisation. S'il s'agit du bloc N, alors celui-ci sera chiffré avec le vecteur d'initialisation chiffré N fois. Le décryptage est très proche du CFB, il faut juste prendre le déchiffré du vecteur d'initialisation.

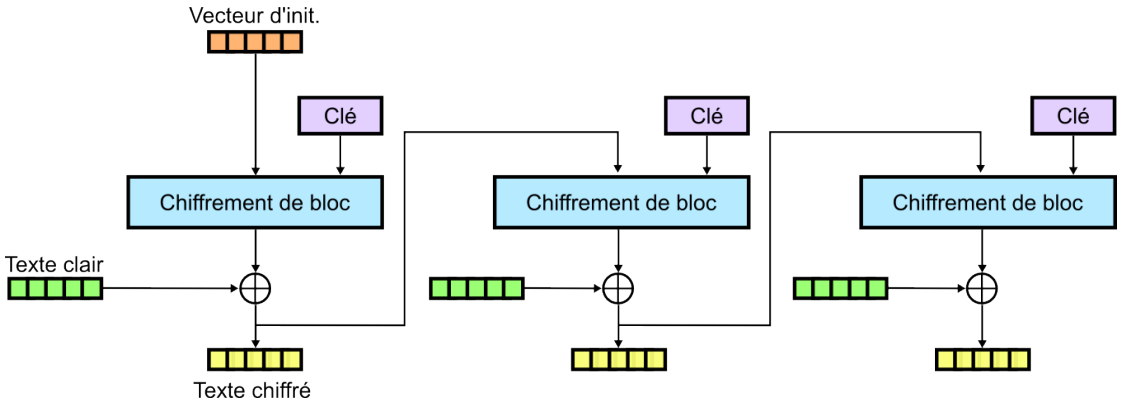

FIGURE 2.6 – schema OFB - Chiffrement

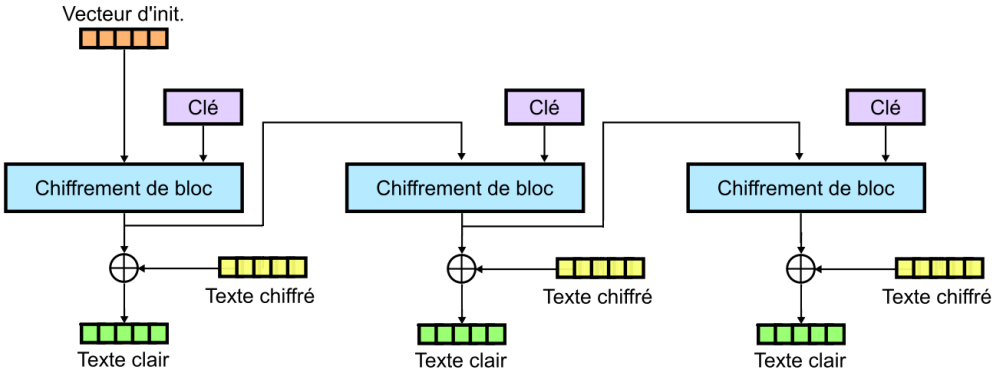

FIGURE 2.7 – schema OFB - Déhiffrement

CTR

Comme le mode OFB, le mode CTR permet le chiffrement par flot et est pré-calculable. De plus il offre un accès aléatoire aux données, est parallélisable et n'utilise que la fonction de chiffrement.

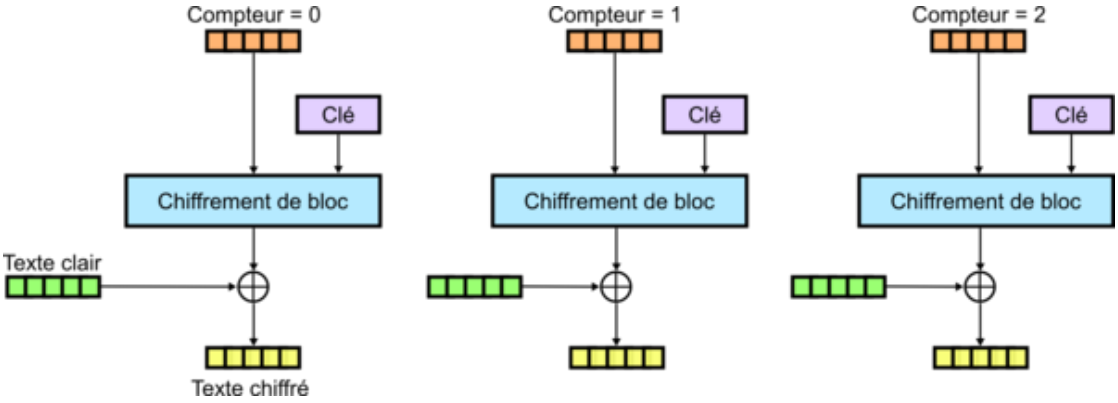

FIGURE 2.8 – schema CTR - Chiffrement

CHAPITRE 3

Les alternatives à AES

Nous discuterons dans ce chapitre des alternatives existantes à l'AES.

Rijndael - Twofish - Serpent

Rijndael est l'algorithe qui remporta en octobre 2000 le concours AES lancé en 1997 par NIST et devient le nouveau standard de chiffrement. (Advanced Encryption Standard)

Parmis les finalistes, autre que Rijndael, se trouvaient les algorithmes Serpent et Twofish. Ces deux algorithmes sont des algorithmes de chiffrement par bloc qui ont subi de nombreuses analyses cryptographiques. Cependant aucun de ces deux algorithmes n'a reçu beaucoup d'attention depuis l'adoption de Rijndael comme nouveau standard. Les derniers résultats les plus significatifs datent ainsi de 2000. Rijndael s'est imposé comme nouveau standard et est massivement utilisé.

Salsa20

Salsa20 est un algorithme de chiffrement par flots, contrairement à AES qui est un algorithme de chiffrement par blocs. Cette distinction est importante car un algorithme de chiffrement par flots permet de produire une chaine pseudo-aléatoire de bits auxquels sont appliqués un XOR avec le message à encrypter. Les algorithmes de chiffrement par bloc peuvent être configurés en chiffrement par flots (mode CTR ou OFB).

Comme nous l'avons vu précédement, les algorithmes de chiffrement par blocs peuvent être configurés pour à la fois assurer l'encryption mais également l'authentification. De plus dans la plupart des modes de fonctionnement des algorithmes par blocs on peut accèder à des parties spécifiques du texte crypté. Salsa20 permet un accès à une partie spécifique de la sortie encryptée. En effet l'algorithme salsa20 a besoin d'une clef, d'un vecteur d'initialisation et d'un numéro de bloc, ce qui lui permet de décrypter n'importe quel bloc.

Il est également possible d'utiliser Salsa20 en mode authentifié, pour cela il faut coupler l'algorithme Salsa20 avec un algorithme d'authentification de messages comme le Poly1305 créé par Daniel J. Bernstein. Le problème est que cet usage n'est pas standardisé.

Un des arguments positifs de Salsa20 est qu'il est rapide en logiciel. L'initialisation et le paramètrage de la clef est négligeable, et il possède un faible nombre de cycles par byte par rapport aux autres algorithmes. En effet Salsa20 peut être 2 à 3 fois plus rapide que AES en mode CTR.

Les principales limitations d'un algorithme comme Salsa20 sont les mêmes que pour tous les algorithmes de chiffrement non-standard : ils sont alternatifs et n'ont ainsi pas la même attention et ne sont donc pas standardisés par des organismes comme NIST.

Bien que Salsa20 ait subit un nombre décent d'analyses crypatanalogiques, la plus part positives, ce n'est rien comparé à AES.

Threefish

Threefish est une contribution récente aux algorithmes alternatifs à AES. Threefish fait partie des algorithmes de chiffrement à avoir réussi la plupart des compétitions organisées par NIST. Threefish est un algorithme de chiffrement par blocs qui peut être configuré pour fonctionner sur des blocs de taille 256, 512 ou 1024 bits. Alors que Threefish a subit quelques cryptanalyses, cela reste encore relativement limité. Aucun de ces travaux n'ont montré de résultats probants quant à une éventuelle faille de sécurité, ce qui inspire plutôt confiance. Encore une fois, les études menées n'ont rien de comparable avec celle faites sur AES, et ainsi il est difficle de dire où se situe Threefish par rapport à AES en matière de sécurité.

Conclusion

Table des figures