

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
9 mai 2016

AES-GCM

RIGAUD MICHAËL et BADIER CHARLIE

Table des matières

Table des matières	1
Introduction	2
1 Différents modes de fonctionnement cryptographique	3
1.1 ECB	3
1.2 CBC	4
1.3 CFB	4
1.4 OFB	5
1.5 CTR	5
2 Fonctionnement	7
2.1 GCM	7
2.2 GMAC	12
3 Comparaison GCM - CCM - OCB	13
3.1 CCM	13
3.2 OCB	14
3.3 Comparaison des performances	14
Conclusion	16
A Les alternatives à AES	18
A.1 Rijndael - Twofish - Serpent	18
A.2 Salsa20	18
A.3 Threefish	19
Table des figures	20

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 garantit l'intégrité et l'authenticité des données. 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 des différents modes de cryptographie couramment utilisés. Puis, nous expliquerons le fonctionnement de AES-GCM ainsi que de ses autres modes. Enfin nous le comparerons à d'autres algorithmes semblables en termes de complexité.

Différents modes de fonctionnement cryptographique

Il existe plusieurs modes de fonctionnement cryptographiques qui peuvent être mis en oeuvre avec l'algorithme de chiffrement AES. Nous verrons dans un premier temps les principaux modes de fonctionnement sur lesquels nous nous appuierons tout au long de ce rapport.

1.1 ECB

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 indépendamment 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 indépendamment les uns des autres.

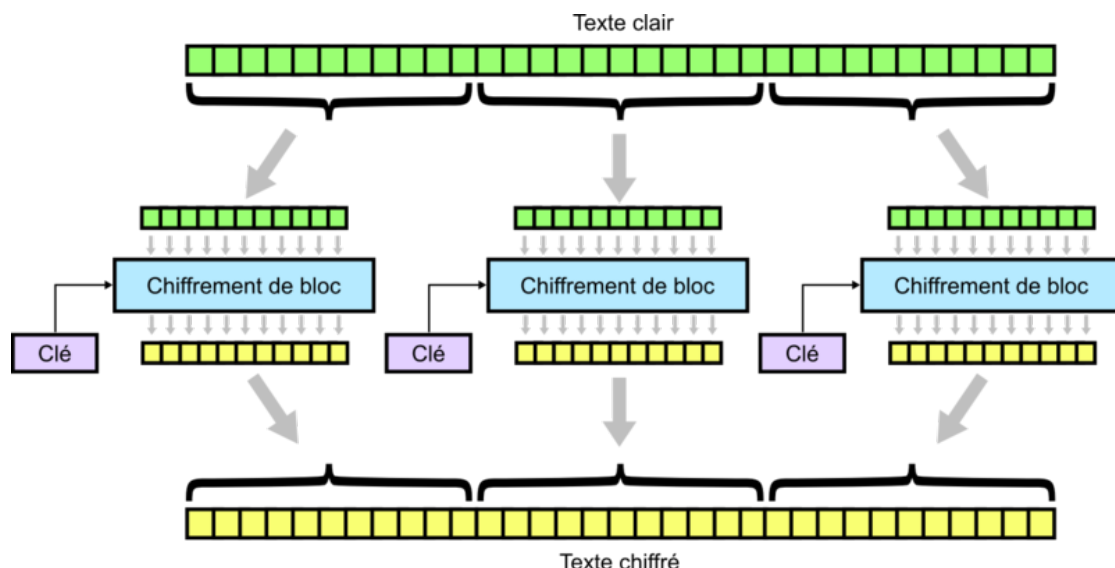


FIGURE 1.1: schéma ECB [?]

Ce mode présente 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 chiffré, permettant ainsi de retrouver le texte clair. Pour ces raisons l'utilisation de ce mode est fortement déconseillé bien qu'il soit défini par défaut dans de nombreuses applications comme MySQL.

1.2 CBC

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

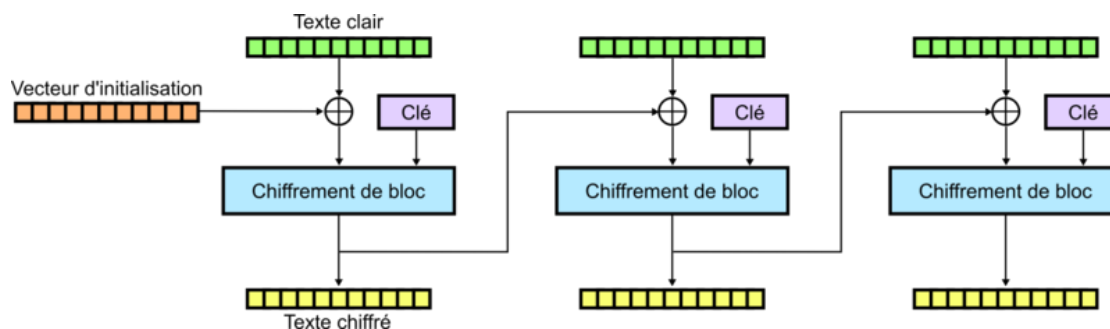


FIGURE 1.2: schema CBC - Chiffrement [?]

Ce mode 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.

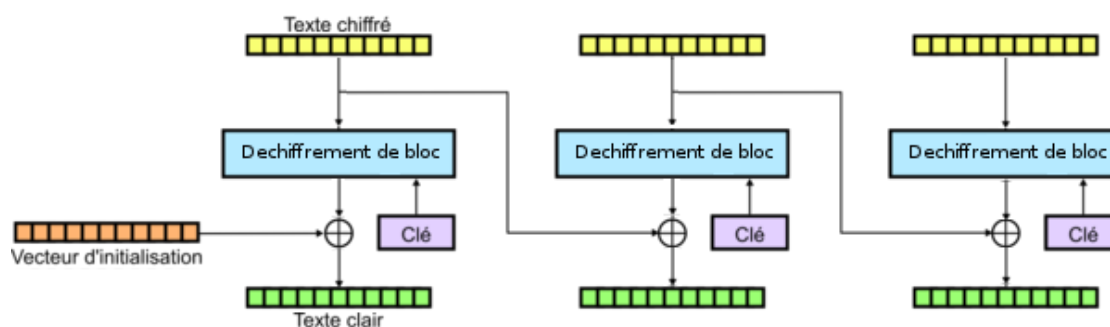


FIGURE 1.3: schema CBC - Déchiffrement

1.3 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 parallélisé. Le déchiffrement est similaire au CBC et peut, quant à lui, être parallélisé.

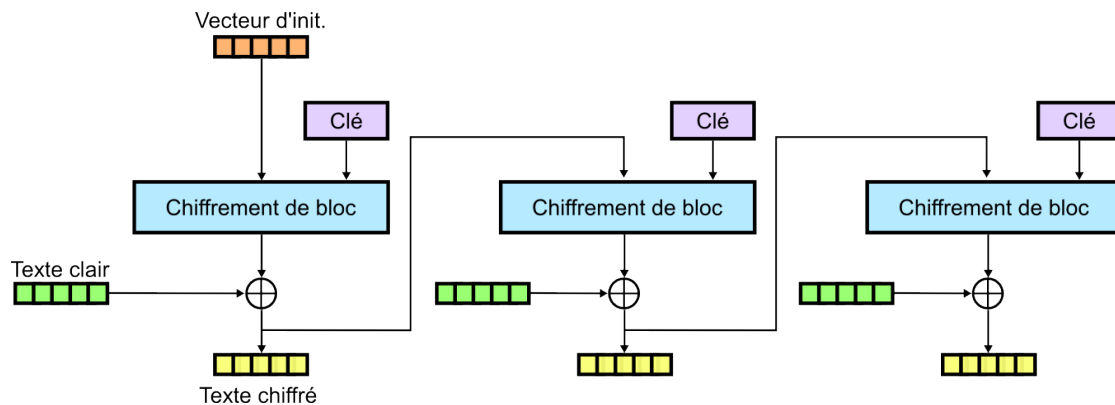


FIGURE 1.4: schema CFB - Chiffrement

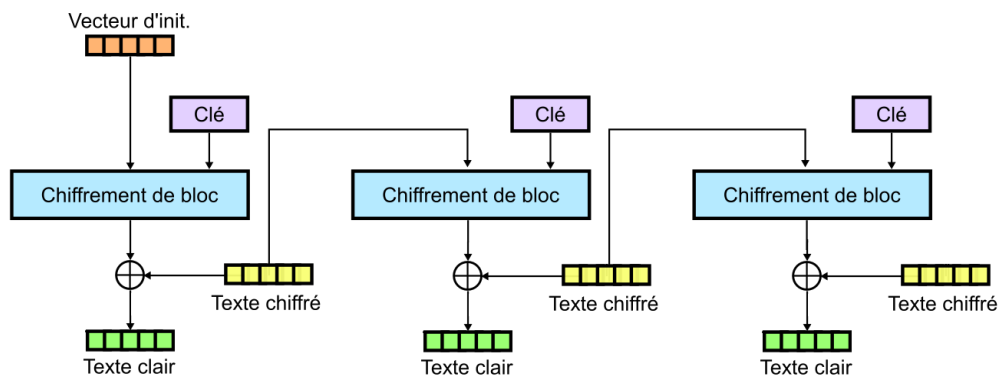


FIGURE 1.5: schema CFB - Déchiffrement

1.4 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.

1.5 CTR

Comme on peut le voir sur la figure 1.8, le mode CTR 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.

Comme le mode OFB, le mode CTR 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.

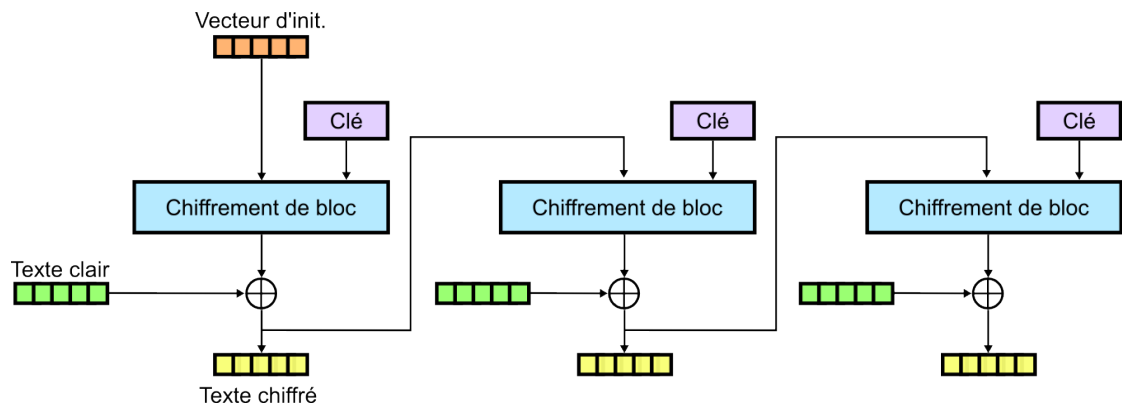


FIGURE 1.6: schema OFB - Chiffrement

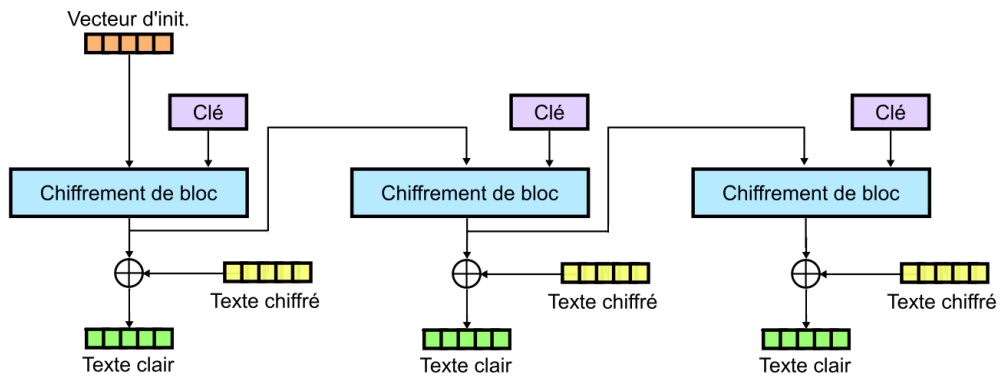


FIGURE 1.7: schema OFB - Déchiffrement

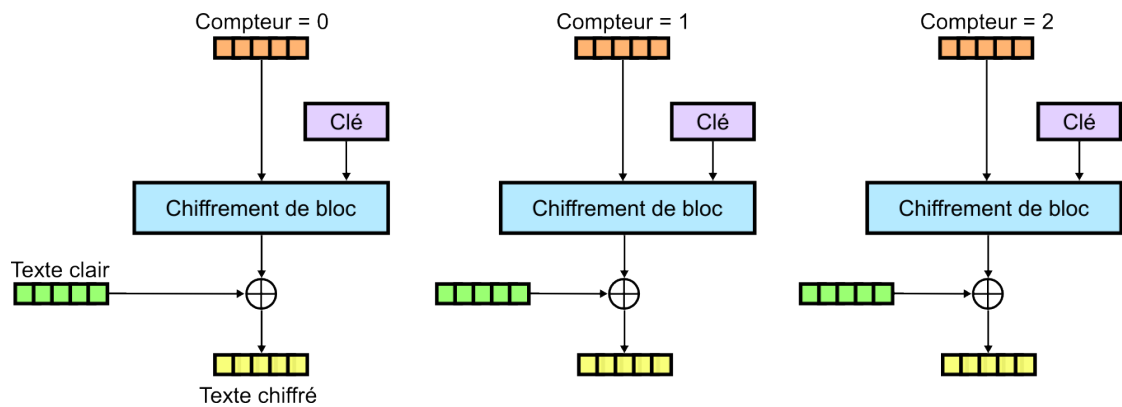


FIGURE 1.8: schema CTR - Chiffrement [?]

Fonctionnement

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

2.1 GCM

Avantages 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.

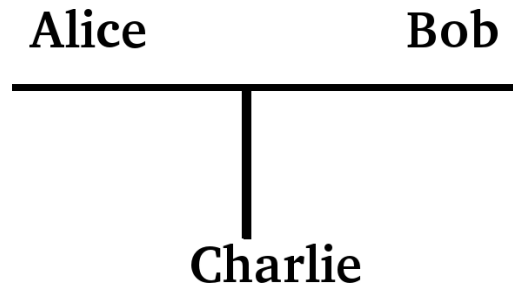


FIGURE 2.1: Bob et Alice

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 aléatoire
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'authentification¹-intégrité².

Dans un premier temps on va parler du bloc de chiffrement. Dans GCM il y a C pour « counter », c'est-à-dire que AES-GCM s'appuie sur un mode qu'on nomme CTR³ qu'on peut voir à la figure 2.2.

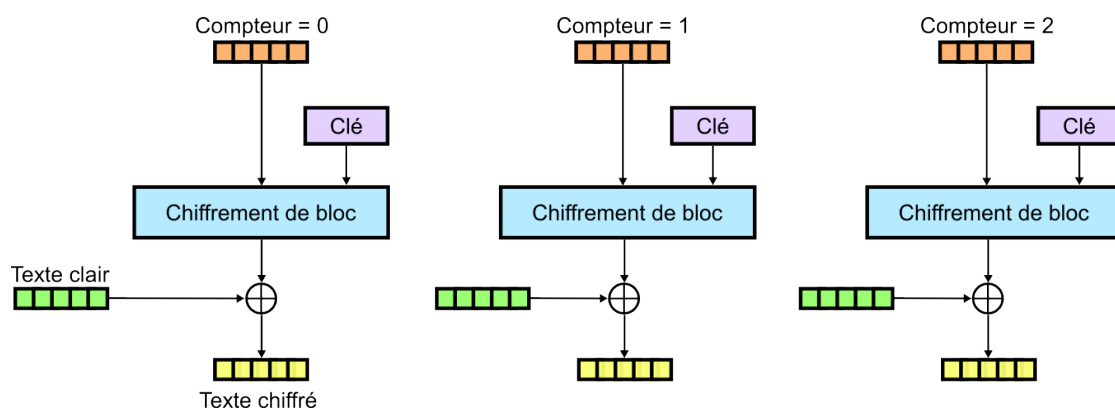


FIGURE 2.2: schéma CTR - Chiffrement [?]

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 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

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. Le mode CTR a été décrit à la page 5, nous rappelons ici que le schéma de fonctionnement.

réalise un XOR du résultat avec le bloc de chiffré suivant. Enfin on refait la multiplication comme on peut le voir sur l'image 2.3.

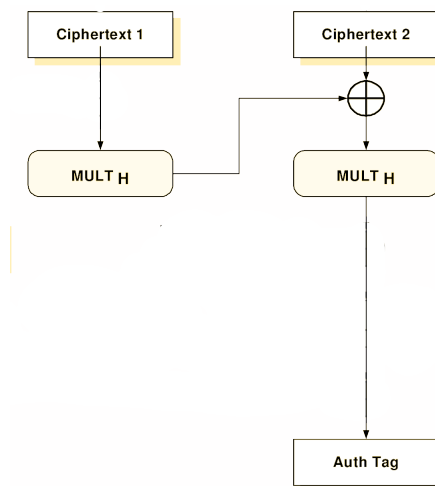


FIGURE 2.3: authentification du message chiffré

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 2.4.

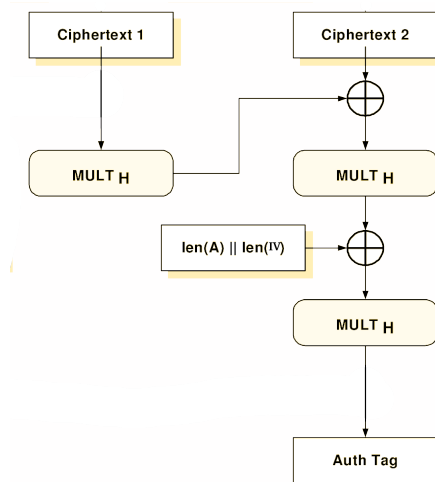


FIGURE 2.4: authentification de la longueur du message et de IV

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 2.5.

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 chiffons 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 2.6.

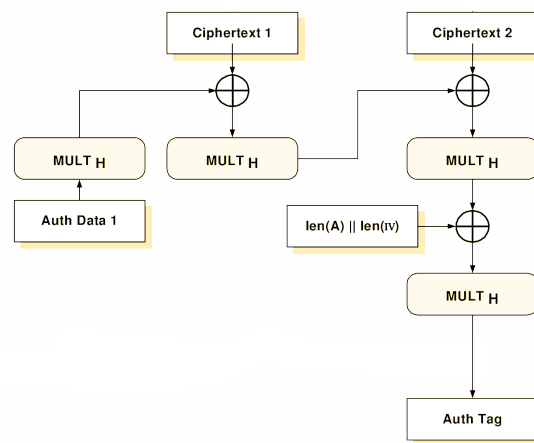


FIGURE 2.5: authentication de data supplémentaire

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

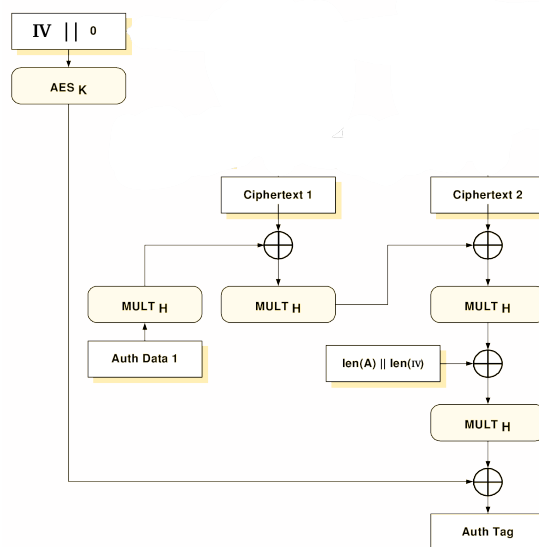


FIGURE 2.6: authentication de la clef K

Le fonctionnement global de AES-GCM est résumé sur l'image 2.7.

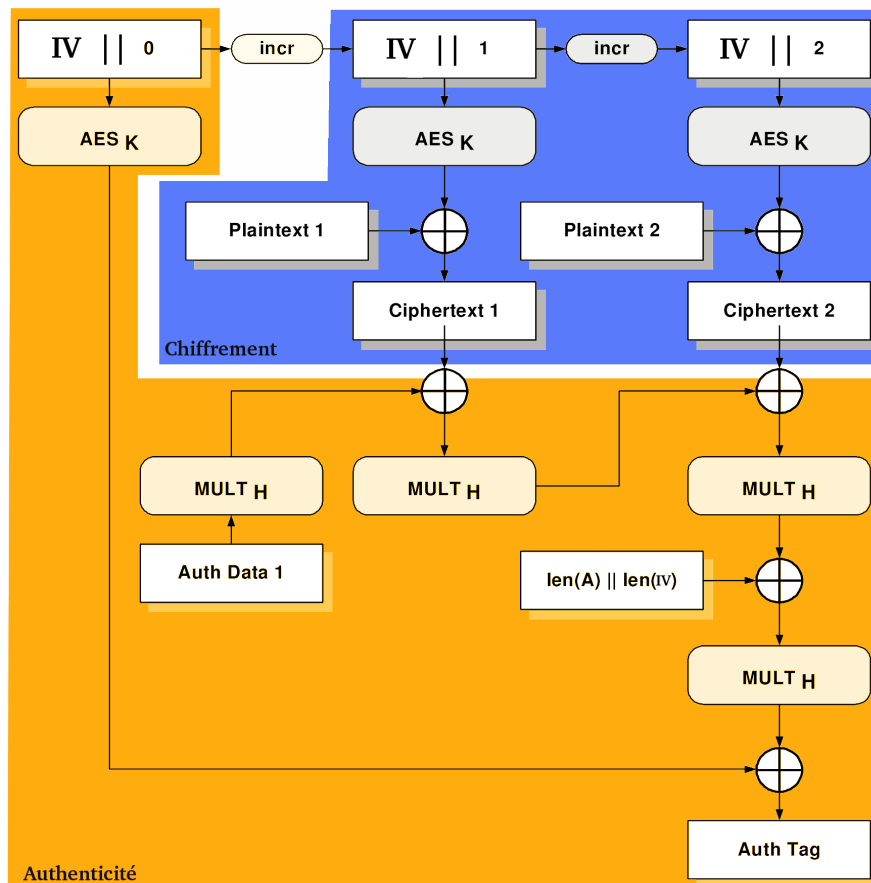


FIGURE 2.7: Fonctionnement de GCM

Déchiffrement

Pour le déchiffrement on utilise exactement le bloc de chiffrement pour obtenir le texte clair et on utilise le chiffré pour obtenir le tag d'authentification. Si on obtient le même tag que celui qui été envoyé alors on est sur d'avoir un message authentifié et identifié.

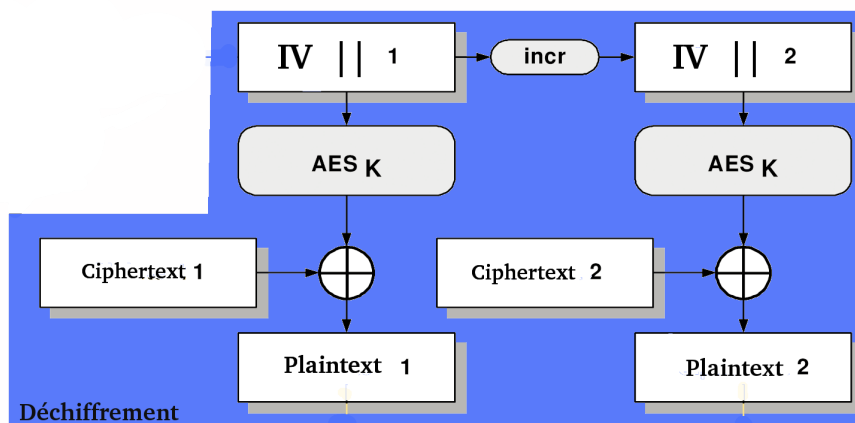


FIGURE 2.8: Fonctionnement du déchiffrement

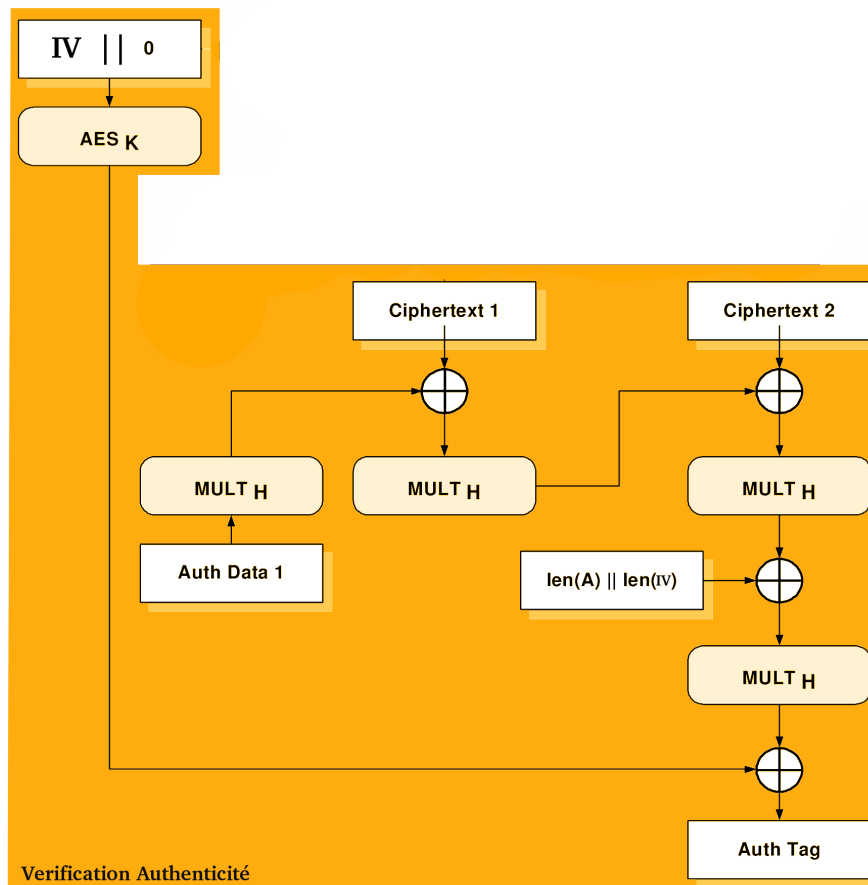


FIGURE 2.9: Fonctionnement de la vérification d'authenticité

2.2 GMAC

GMAC est un cas particulier de GCM où aucun texte brut n'est présenté. C'est donc une variante de GCM qui ne réalise que l'authentification.

Comparaison GCM - CCM - OCB

Lors de nos recherches nous avons constaté que les algorithmes fournissant à la fois la confidentialité, l'authenticité et l'intégrité sont : GCM, CCM, OCB, CWC, EAX et APM.

Mais parmi eux seulement trois sortent du lot : GCM, OCB et CCM. C'est pourquoi nous avons décidé d'écarter les autres algorithmes et de seulement comparer ces trois.

3.1 CCM

Comme son nom le suggère le mode CCM combine le mode CTR et le mode CBC-MAC. Ces deux modes "primitifs" sont combinés pour authentifier les données puis les encrypter. CBC-MAC permet dans un premier temps d'obtenir un tag d'authentification du message clair. Puis le message et le tag sont chiffrés en utilisant le mode CTR.

Le vecteur d'initialisation (IV) doit être choisi avec précaution car il ne doit jamais être utilisé plus d'une fois par clef. En effet le mode CCM est un dérivé du mode CTR.

Une idée essentielle est que la même clé de cryptage peut être utilisée à la fois pour l'authentification et l'encryptage, à condition que les valeurs de comptage utilisées dans le cryptage ne rentrent pas en collision avec le vecteur d'initialisation (IV) utilisé pour l'authentification.

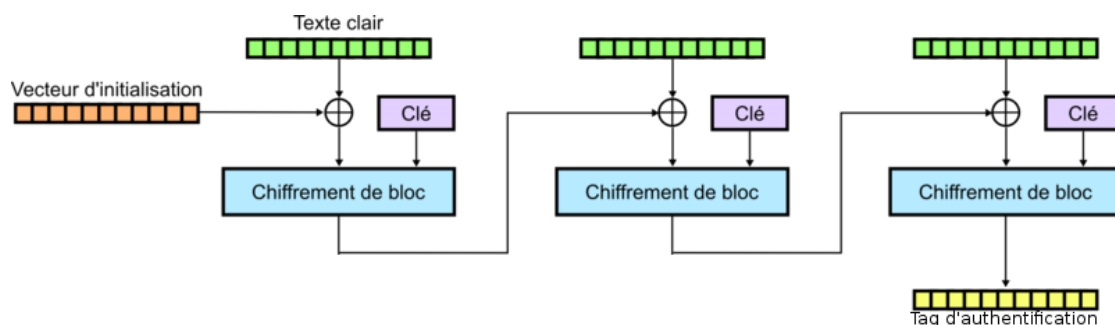


FIGURE 3.1: Création du tag d'authentification avec CBC-MAC

Contrairement à AES-GCM, la génération du tag d'authentification (via CBC-MAC) est faite à partir du message en clair et non pas à partir du message encrypté. Au niveau performance, le mode CCM va utiliser de manière générale plus de calculs que AES-GCM. En effet, dans AES-GCM l'authentification et l'encryption ne font appel qu'une seule fois à l'algorithme AES par bloc, tandis que, le mode CCM va utiliser une première fois le chiffrement AES pour l'encryption puis une seconde fois pour générer le tag d'authentification. De plus, le mode CCM ne peut pas être parallélisé.

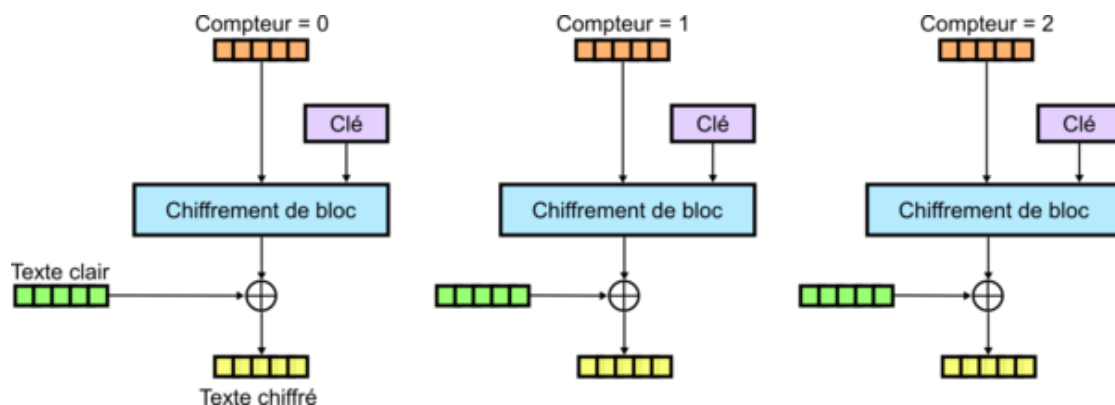


FIGURE 3.2: Chiffrement du tag et du message avec CTR

3.2 OCB

Le mode OCB est lui aussi conçu pour fournir à la fois l'authentification et la confidentialité. OCB (Offset CodeBook) est basé sur le mode ECB avec l'utilisation d'un vecteur d'initialisation. Pour l'authentification il faut d'abord effectuer un checksum du message clair, ce checksum est ensuite encrypté comme un block du message clair.

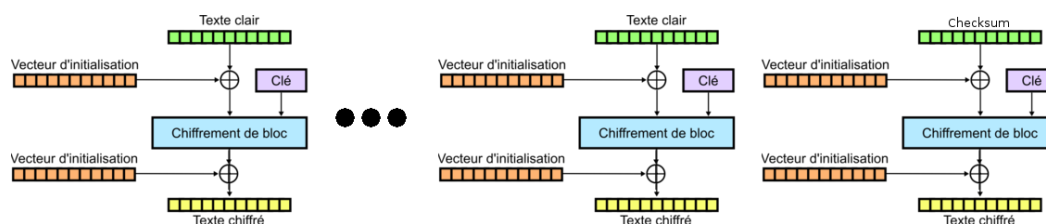


FIGURE 3.3: Fonctionnement du mode OCB

AES-OCB est un mode de fonctionnement qui est soumis à deux brevets aux Etats-Unis qui empêchent son utilisation dans toute application commerciale ou gouvernementale aux Etats-Unis. De plus Niels Ferguson a montré l'existence d'attaques sur le mode OCB qui limitent l'envoi de données à 64GB par clef. D'un point de vue performance, le mode OCB semble plus rapide que AES-GCM car il ne nécessite pas l'implémentation des blocs de multiplications dans l'espace de Galois.

3.3 Comparaison des performances

comparaison extrait d'une étude

Pour obtenir les performances de ces modes nous nous sommes appuyés sur le travail de Ted KROVETZ et Phillip ROGAWAY [?].

Sur la figure 3.4, il est possible d'observer la rapidité de modes OCB, CCM et GCM sur des architectures différentes. Sur l'axe des abscisses on peut observer la taille du message en bytes, et sur les ordonnées le nombre de cycles par byte.

On remarque que sur la totalité des architectures testées OCB est plus performant que ses rivaux.

Cette étude confirme ce que nous avons révélé précédemment, c'est-à-dire que OCB est plus performant que AES-GCM.

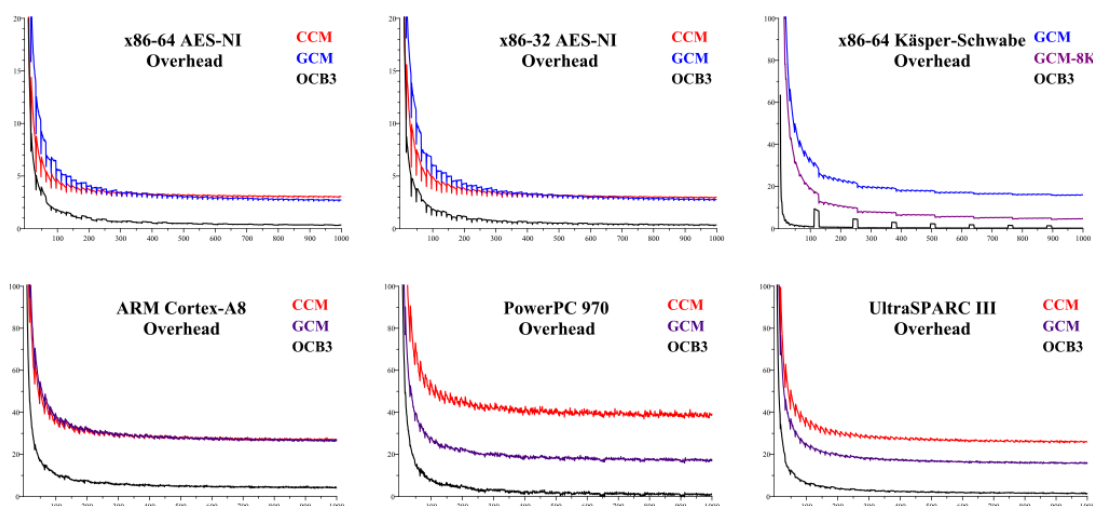


FIGURE 3.4: Performance empirique sur différents architectures[?]

Comparaison réalisé sur nos machines personnelles

Sur nos machines (Asus N76VB avec Ubuntu 14.04) nous avons également essayé de vérifier ces performances. Pour cela nous avons d'abord utilisé le site <http://www.faux-texte.com/lorem-ipsu-20.htm> pour générer un faux texte pseudo-aléatoire de 6.8M¹. Ensuite, nous avons exécuté la commande suivante dans notre terminal linux :

```
1 time openssl enc -e -in "fichier_test" -aes-256-gcm -out "fichier_encrypt" -k "toto"
```

Voici la sortie que nous avons pu observer :

```
1 openssl enc -e -in "fichier_test" -aes-256-gcm -out "fichier_encrypt" -k
0,02s user 0,01s system 98% cpu 0,027 total
```

Donc notre ordinateur a mis 0.02s pour chiffrer notre fichier.

Malheureusement nous ne pourrons pas comparer cette performance aux mode OCB et CCM puisqu'ils ne sont pas implémentés dans openssl. Néanmoins, nous pouvons constater la rapidité de l'opération qui a été réalisée. Nous supposons que dans une utilisation courante cette rapidité est plus que suffisante pour un utilisateur lambda.

De plus au cours de notre réflexion nous avons envisagé de proposer une alternative à AES-GCM en remplaçant AES par un autre algorithme de chiffrement. Nous pensons néanmoins que cela sort du cadre de notre étude c'est pourquoi il est possible de trouver en annexe à la page 18 les alternatives à AES.

1. Il est possible de visualiser le fichier que nous avons utilisé sur notre github <https://github.com/magichal/aes-gcm>

Conclusion

Face à cette étude nous pouvons conclure que AES-GCM est un algorithme très puissant et rapide qui permet d'assurer la confidentialité, l'intégrité et l'authenticité des données.

Nous avons pu dans un premier temps décrire le fonctionnement de AES-GCM , puis nous l'avons comparé à d'autre algorithme répondant aux mêmes caractéristiques : OCB et CCM. Nous avons remarqué à la fin de cette étude de comparaison que GCM n'était pas l'algorithme le plus rapide puisque sur plusieurs architectures différentes c'est OCB qui se distingue. Néanmoins OCB est un algorithme sous brevet au États-Unis et qu'il est sensible à un certain nombre d'attaques.

De plus, comme nous avons pu le souligner à travers des tests sur nos machines, AES-GCM est un algorithme relativement performant. Nous pensons qu'il est suffisant pour une utilisation personnelle voir professionnelle.

Annexe

Les alternatives à AES

Nous discuterons dans ce chapitre des alternatives existantes à l'AES. Ces alternatives sont pertinentes car elles pourraient améliorer les performances de AES-GCM sans réduire sa capacité d'authentification et d'intégrité.

A.1 Rijndael - Twofish - Serpent

Rijndael est l'algorithme 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.

A.2 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 chaîne 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 précédemment, 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 accéder 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 clé, 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 paramétrage de la clé 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 subi un nombre décent d'analyses cryptanalytiques, la plus part positives, ce n'est rien comparé à AES.

A.3 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 subi 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 difficile de dire où se situe Threefish par rapport à AES en matière de sécurité.

Table des figures

1.1	schéma ECB [?]	3
1.2	schema CBC - Chiffrement [?]	4
1.3	schema CBC - Déchiffrement	4
1.4	schema CFB - Chiffrement	5
1.5	schema CFB - Déchiffrement	5
1.6	schema OFB - Chiffrement	6
1.7	schema OFB - Déchiffrement	6
1.8	schema CTR - Chiffrement [?]	6
2.1	Bob et Alice	7
2.2	schéma CTR - Chiffrement [?]	8
2.3	authentification du message chiffré	9
2.4	authentification de la longueur du message et de IV	9
2.5	authentification de data supplémentaire	10
2.6	authentification de la clef K	10
2.7	Fonctionnement de GCM	11
2.8	Fonctionnement du déchiffrement	11
2.9	Fonctionnement de la vérification d'authenticité	12
3.1	Création du tag d'authentification avec CBC-MAC	13
3.2	Chiffrement du tag et du message avec CTR	14
3.3	Fonctionnement du mode OCB	14
3.4	Performance empirique sur différents architectures[?]	15