REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE M'HAMED BOUGARA – BOUMERDES

Faculté de Technologie

Département Ingénierie des Systèmes Electriques

Mémoire de Master

Présenté par

BENTRAD ABDERRAOUF et REGUIEG YSSAAD

Filière : Télécommunication

Spécialité : Réseaux et Télécommunication

ANALYSE ET SIMULATION DU CODE DETECTEUR ET CORRECTEUR D'ERREUR DE REED-SOLOMON

Soutenu le ……. / ……. /2020 devant le jury composé de:

-	RIAHLA	Mohamed amine	MCA	UMBB	President
-	MESSAOUDI	Noureddine	MCA	UMBB	Examinateur
-	HAMADOUCHE	M'hamed	Prof	UMBB	Promoteur

Année Universitaire : 2019/2020

Remerciement

En préambule, nous souhaitons rendre grâce à Dieu, le clément et le miséricordieux de nous avoir donné la force et la patience de mener à bien ce modeste travail.

Nous tenons évidemment à entamerle remerciement en témoignant de notre profonde reconnaissance envers Monsieur le Professeur HAMADOUCHE M'hamed, pour nous avoir encadré et dirigé ce travail avec patience, ainsi que pour sa riche contribution et ses précieux conseils.

Nous remercions vivement Monsieur MASSAOUDI, chef département, sa patience et son organisation nous ont permis de surmonter de nombreuses difficultés survenues durant la formation

Nous tenons aussi à remercier toute l'équipe pédagogique (les professeurs et nos camarades d'étudiants, pour nous avoir transmis leur savoir tout au long de notre cycle d'étude, et qui nous ont prêté main forte.

Notre gratitude s'adresse également à tous ceux qui, de loin ou de prés, ont participé à la réalisation de ce travail.

Nous voulons exprimer nos remerciements aux membres jurys

Nos remerciements s'adressent aussi, à tous ceux qui nous ont prêté main forte.

Dédicace

Tout d'abord, je remercie **ALLAH** pour la faveur de la santé et de L'Islam et le tout puissant de m'avoir donné le courage pour accomplir ce modeste travail que

Je dédie :

A mes parents, ma mère et mon père, l'être les plus chers au monde qui n'ont jamais cessé de prier pour moi et pour tous leurs affections et leurs amours que dieu les protège inchallah.

A ma femme et mes enfants, pour tous ses encouragements, ses sacrifices.

A mon binôme Abderraouf, Ali, Islam, et Massi et pour tous les bons moments qu'on a passés ensemble, je vous souhaite beaucoup de réussite dans votre carrière.

A mon encadreur pour sa patience, ses conseils, sa disponibilité et pour l'excellence de son accompagnement et la confiance qu'il nous a accordé.

A mes enseignants sans aucune exception, A mes collègues promotion 2020 et à tous ceux que j'aime et qui m'aime.

Reguieg Yssaad Amine

Dédicace

En tout premier lieu, je remercie **ALLAH**, *tout puissant, de m'avoir donné la force pour survivre, ainsi que l'audace pour dépasser toutes les difficultés*

J'ai le grand honneur de dédier ce travail A l'être le plus cher à mon cœur, Maman

A celui qui a fait de moi un Homme, Mon père.

A ma belle femme

A mon cher fils

Mes chères Sœurs

A tous mes amis

Et mon binôme

A tous mes collègues qui me donnent le courage.

A tous ceux et celles qui m'ont aidé de loin ou de prés.

Je dédie ce modeste travail.

BENTRAD ABDERRAOUF

Sommaire

Liste des figures

Liste des tableaux

La liste des abréviations

Il existe dans la plupart des cas constatés une abréviation en français et d'autres en anglais, donc, toutes les deux sont indiquées une première fois puis nous employons l'abréviation la plus usuelle, c'est en anglais, à cet effet, la signification d'une abréviation ou d'un acronyme n'est souvent indiquée qu'à sa première apparition dans le texte.

A	**ADSL:** Asymmetric Digital Sbscribre Line.
B	**BBAG :** Bruit Blanc Additif Gaussien. **BCH:** Bose-Chaudhuri–Hocqenghem.0 **BER:** Bit Error Rate.
C	**CBC :** Canal Binaire Symétrique. **CAE :** Canal à Effacement. **CD :**Compact Disc. **CDMA:** Code Local Area Network. **CRC:** Cyclic Redundancy Checksum.
D	**DVD:** Digital Versatile Disc. **DDVB:** Digital Video Broadcasting
G	**GF:** Galois Field
L	**LDPC:** Lowe Density Parity Check.
X	**XOR :** Ou exclusif ou « exclusive OR ».
O	**OFDM:** Orthogonale Frequency Division Multiplexing.
P	**PSNR:** Peak Signal to Noise Ratio.
R	**RS:** Reed-Solomon.
V	**VDSL:** Very High Bit-rate Digital Subscriber Line.
W	**WLAN:** Wireless Local Area Network
M	**MCD:** Modèle Conceptuel de Données

Résumé

Les codes correcteurs d'erreurs sont utilisés dans le système de transmission telle que les satellites, la téléphonie, les disques lasers, les téléviseurs haute définition, le traitement d'image et de la parole, cryptographie (signature, carte à puce), compression de données, etc.…,. L'objectif du code correcteur d'erreurs, est permet de transmettre d'une façon fiable, l'information codée au moyen de mots binaires d'une longueur donnée, sur des lignes plus ou moins bruitées. Tel que cette transmission bruitée présente un risque d'erreurs variable selon les cas enregistrés, il s'agit de trouver un moyen et de le corriger à la réception de l'information, au prix d'une certaine redondance, tout en minimisant le temps d'occupation de la ligne dans chaque situation

Ces codes dont la source, ayant un problème très concret lié à la transmission de données, dans la plupart des cas, une transmission de données se fait en utilisant une voie de communication, soit un canal de communication qui n'est pas intitule et non fiable, tant que les données au cours de leur transport dans le canal sont susceptibles d'être altérées. Lors d'une communication radio, la présence de parasites sur la ligne va perturber le signal portant le son et la voix. A cet effet, on veut savoir comment on peut récupérer le signal bruité et les informations originales, quel type d'algorithmes on utilisera pour rendre cette communication plus fiable avec un taux d'erreurs plus correcte.

A la lanterne de ce qui précède on propose dans ce travail une étude détaillée sur le code de correcteur d'erreur « Reed-Solomon » qui est utilisé justement pour limiter les erreurs enregistrées pendants la transmission.

Introduction générale

De nos jours, on vit dans un monde où la communication joue un rôle primordial tant par la place qu'elles occupent dans le quotidien de chacun, que par les enjeux économiques et technologiques dont elles font l'objet. Ce développement rapide des réseaux mondiaux et les immenses possibilités offertes par les transactions électroniques en communications continues, posent aujourd'hui de manière cruciale, le problème de la protection de l'information contre les erreurs de transmission d'une part, et d'autre part, il faut que ces données soit non intelligibles sauf à l'auditoire voulu. Afin de pallier ces deux contraintes on utilise le codage de l'information pour combattre les erreurs de transmissions.[1]

C'est dans la course au débit et à la fiabilité que les codes correcteurs entrent en jeux, dont la communication avec les sondes spatiales, à l'autre bout du système solaire, etc…pose le problème de la fiabilité du message transmis, sur une telle distance est obligatoirement parasitée (notamment à cause de diverses sources de perturbations électromagnétiques). Tous en précisant aussi les supports de communication multiples et chacun présente de différentes performances, aussi, les techniques d'accès à ces supports sont divers et sont intégrées pour gérer l'usage du canal de transmission à travers un algorithme bien étudié, en particuliers les techniques à accès aléatoire qui présentent jusqu'à nos jours des taux d'erreurs importants, d'où, il est important de « corriger » cette transmission, dont lequel, c'est le rôle de code d'erreur appelé « Reed-Solomon » pour la sécurisation de la donnée, c l'un des codes où on rajoute au message à transmettre des informations supplémentaires, permet de reconstituer le message au niveau du récepteur, de corriger une ou plusieurs erreurs dans un mot de code en ajoutant aux informations des symboles redondants, autrement dits, des symboles de contrôle.[2]

Or, les codes correcteurs d'erreurs dont le code Reed-Solomon, est un mécanisme ou un processus qui permet de corriger les erreurs dans un mot de code en ajoutant aux informations des symboles redondants qu'on appelle des outils de contrôle, comme on a expliqué supra, Ce code est utilisé pour accroitre la fiabilité de système de transmission et augmenter leur débit [10]. Alors, on va présenter au début de cette thèse, une étude bien détaillée sur les différents codes correcteurs d'erreurs, notamment le code de Reed-Solomon, à savoir, les opérations de codage et décodage en donnant des exemples pour mieux éclaircir les stratégies et le principe de ces opérations.

Cette thèse est composée de trois chapitres, on va entamer dans le premier chapitre d'introduire les différents concepts des codes correcteurs d'erreurs tel que les codes en blocs et les codes convolutifs, ensuite on va baser d'une manière particulière sur le code de Reed-Solomon dans la deuxième chapitre, et à la fin de la troisième chapitre, on traite par une simulation sur Matlab, pour évaluer les performances de code de Reed-Solomon. A l'issu, on résume ce travail par une conclusion générale.

CHAPITRE 1 :
LES CODES CORRECTEURS D'ERREURS (GENERALITES)

1.1 Introduction

La conception des codes est une discipline des mathématiques très utilisées, dont le sujet est la transmission d'informations sur un canal de transmission bruité, en utilisant des objets combinatoires et algorithmiques nommés codes correcteurs d'erreurs. Pour introduire ce sujet, il est nécessaire de détailler les notions de base du codage et de connaitre le chemin qu'empreinte un message depuis l'émetteur jusqu'au destinataire, et de considérer les concepts intéressants qui apparaissent. Il existe trois étapes impliquées dans la chaine de transmission: la source, le récepteur et le support de transmission.

L'objectif de la conception des codes correcteurs d'erreurs est de créer de codes capables de détecter et de corriger des erreurs arrivées lors de la transmission d'un message, à la base théorique de la conception de l'information, commencé par l'article de Claude Shannon (1948) [3].

Les codes correcteurs sont largement utilisés dans le domaine des télécommunications, où ils permettent des transmissions fiables sur des médiums de communication bruités comme les canaux sans fil. On les retrouve également dans le domaine du stockage, pour protéger l'information enregistrée face à la détérioration du support [10].

Dans ce chapitre, on va représenter les deux grandes familles des codes correcteurs d'erreurs, c'est les codes convolutifs et les codes en bloc de différente architecture.

1.2 La transmission numérique

La transmission numérique se déplacent de l'information entre une source et un destinataire, tous en utilisant un support physique, à titre exemple, le câble, la fibre optique ou encore la propagation sur un canal radioélectrique. Les signaux transportés sont d'origine numérique comme dans les réseaux de données, ou analogique (voix, parole, une image...etc.), qui est convertis sous une forme numérique. Le principe de ce système de transmission est de circuler l'information de la source vers le destinataire avec plus de fiabilité [1].

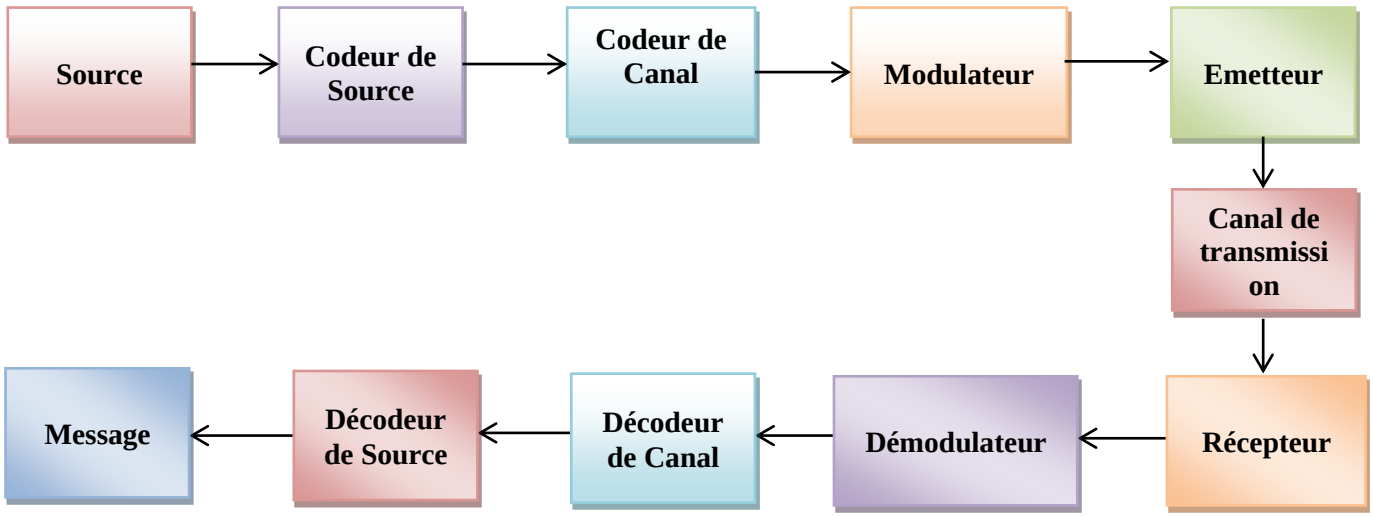


Figure 1.1: *Schéma bloc d'une Transmission numérique* [2].

On trouve des différents blocs modélisant les traitements successifs apportés à l'information dans la chaine de transmission numérique, d'où, les blocs peuvent être énumères comme suit [2] :

Source:	Elle émet un message numérique sous forme d'une suite d'éléments binaires par Bits
Codage de source:	Il supprime les redondances dans le message pour rendre les éléments binaires mutuellement indépendants
Codage de canal:	Il insère des éléments binaires afin d'améliorer la qualité de la transmission ;
Modulateur:	Il traduit le message binaire en signal en le transportant dans les éléments tel que l'air, l'eau, les câbles etc
Emetteur:	Il permet au signal de se propager dans le canal de transmission
Récepteur:	Il capte le signal émis
Démodulateur:	Il traduit le message reçu en signal binaire
Décodeur de canal:	Il détecte et corrige les erreurs de transmission suite aux éléments binaires ajoutés lors du codage
Décodeur de source:	Il régénère le message binaire

1.3 Les canaux

Afin d'échanger et circuler l'information d'un point à un autre, les systèmes de communication utilisent un support de transmission, qu'on appelle aussi« Canal de transmission », il introduit des perturbations, affaiblissements, écho, et bruit qui détériorent l'information émise et créent des erreurs dans le message. Pour plus de fiabiliser de message, c à dire, empêcher la perte due aux perturbations, bruit, ces systèmes intègrent un processus de protection du message émis., dont le principe général de cette protection est l'ajout de redondance, pour avoir une information supplémentaire, plus optimale possible en termes de coût, de volume et de contraintes, dépendant largement du canal [1].

On présente maintenant les trois types de canal :

1.3.1 « CBS » Canal Binaire Symétrique

Il est caractérisé par la probabilité d'erreur ***p*** au cours de la transmission un bit (**0** ou **1**) qui modifiée son opposé. Ces modifications se produisent indépendamment sur chacun des bits transmis.

Le canal CBS peut être représenté par le schéma suivant [1] :

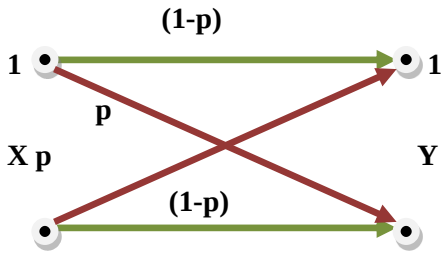


Figure 1.2** : Canal binaire symétrique **[4].

1.3.2 Le canal à effacement(CAE)

Ce canal (CAE) est discret, stationnaire et sans effet mémoire [3], tout comme le CBS, dont les erreurs interviennent sur ce type de canal sont des effacements des informations.

Par contre, au canal binaire symétrique, l'information transmise sur ce canal n'est pas altérée, mais une partie de celle-ci est tout simplement perdue. Sur ce canal de paramètre p ; la probabilité de symbole transmis soit effacé est égale à p, modélise souvent ce canal en ajoutant à l'ensemble des valeurs pour prendre la sortie du canal, un symbole E représentant l'effacement comme indiqué dans la figure (1.3).

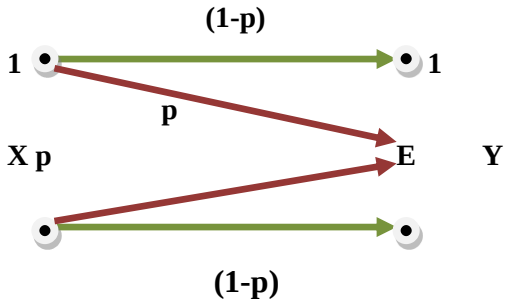


***Figure1.3** : Représentation d'un canal binaire à effacements [4].*

1.3.3 Canal à bruit additif blanc gaussien

C'est le modèle de canal le plus fréquemment utilisé pour la simulation des transmissions numériques, il est aussi un des plus faciles à générer et à analyser ; c'est le Canal à Bruit Blanc Additif Gaussien (BBAG). Ce bruit modélise à la fois les bruits d'origine interne (bruit thermique dû aux imperfections des équipements…) et le bruit d'origine externe (bruit d'antenne…). Ce modèle est souvent associé à une transmission filaire, puisqu'il représente une transmission quasi-parfaite de l'émetteur au récepteur.

Le signal reçu s'écrit alors : (t)= s(t)+v(t) **(1.1)**

Où v(t) représente le BBAG est un bruit dont la densité spectrale de puissance est la même pour toutes les fréquences (bruit blanc). Il est dit additif car il est simplement ajouté au signal entrant. Enfin, il est dit gaussien du fait de sa densité de probabilité de transmission définie comme suit [1,3]:

$$p(r|\ s) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(r-s)^2}{2\sigma^2}} \quad \textbf{(1.2)}$$

ou :

σ: La variance.

r: Le signal reçu.

s : Le signal transmis.

1.4 Les codes correcteurs d'erreurs

L'objectif des codes correcteurs d'erreurs, permettra la transmission d'information malgré l'ajout éventuel d'erreurs lors de la transmission, avec des techniques de codage de l'information basée sur la redondance du message à transmettre [5].

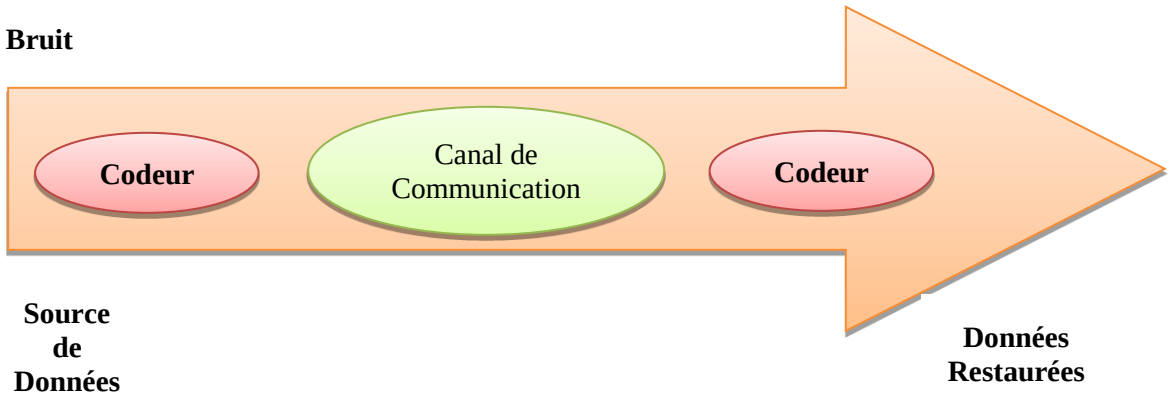


Figure1.4 : *Schéma général des Codes Correcteurs.*

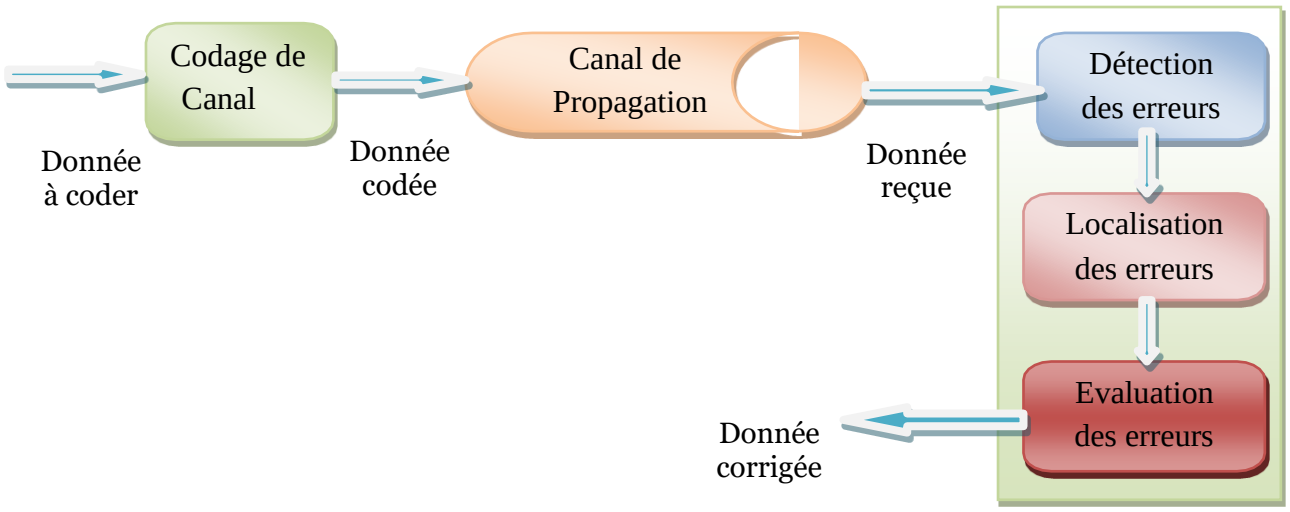


Figure1.5 : *l'utilisation d'un code correcteur d'erreurs [21].*

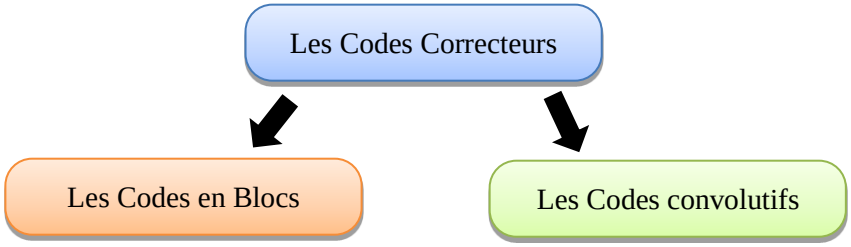


***Figure1.6* :** *les familles des codes correcteurs d'erreur*

1.4.1 Les codes en blocs

Le fonctionnement décodeur dans un code en bloc divise la séquence d'information en bloc de message de taille fixe k bit, pour transformer chaque message Di en un mot de code Ci de taille n en appliquant la loi linaire, d'où, la redondance associée à chaque bloc est de taille r, dont k + r = n, il est défini par la formule suivante :

$$R = \frac{k}{n} \qquad \textbf{(1.3)}$$

ù, k et n représentent respectivement les nombres de bits en entrée et en sortie du codeur,

Le taux de codage est : (n-k).

Les codes en blocs linéaires constituent un faible pourcentage de l'ensemble des codes en blocs ; qui sont les plus utilisés dans la pratique [7,10].

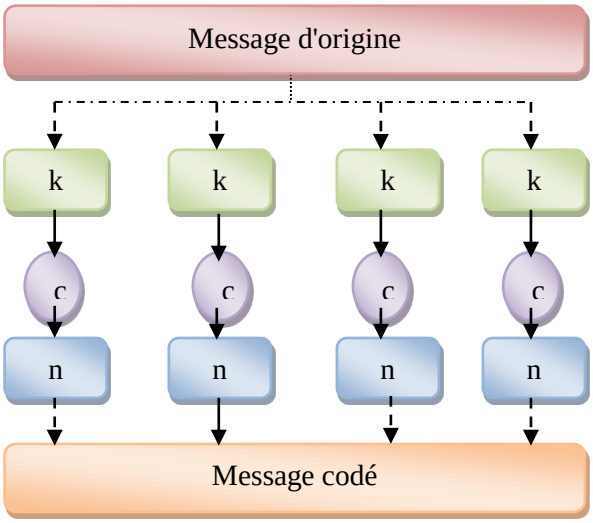


Figure1.7: *Codage en bloc* [11].

Nous donnons ci-dessous une liste de quelques familles de codes en bloc linéaires.

1.4.1.1 Les codes à répétition

Ce code consiste à répéter *n* fois le bit d'information : [*n ; 1 ; d*], dimension « 1 », longueur « *n* » et distance « *d* ». Pour « *n* »impaire, ce code est parfait et optimal lorsque l'on protège un seul bit de donné, à titre exemple, pour *n* = 3 :

Figure1.8 : *codage de code à répétition.*

On constate que ce code est de dimension « 1 » et de longueur « n », et chaque bit étant pris un par un, transformé en « n » bits, donc, le rendement est de «$\frac{1}{n}$ », ce qui est très faible, la distance minimale entre deux mots de code est «n », donc, il faut changer les « n » répétitions d'un bit de mot de code pour obtenir un autre mot de code, qui permet de corriger jusqu'à [$\frac{1}{n}$] erreur, qui dispose d'un algorithme de décodage très simple : il suffit de prendre le bit majoritaire de chaque bloc de « n » bits, si l'on reprend l'exemple: n = 4, le mot reçu est: 0100, le mot de code le plus proche est 0000 et le mot d'information qui a été envoyé est « 0 ».

On constate dans cet exemple, si le mot reçu est : 0110 alors, il existe deux mots de code à distance 2 ; on détecte toujours la présence d'une erreur mais le décodage n'est plus unique, donc, on ne peut pas savoir si le mot envoyé est 0 ou 1.

Figure1.9 : *Le décodage de code à répétition.*

Ce décodage s'effectue par choix majoritaire, dont la capacité de correction est donnée par

$$\frac{n-1}{2}$$

On utilise ce code en télécommunication pour reconnaître une modulation de signal [1,5].

1.4.1.2 Les codes de parité :

Ce code à la transmission des caractères de texte ou un signe, il est conventionnellement une suite de 7 chiffres binaire, exprimé sous forme binaire, découpé en mots d'information qui sont des caractères. Ce codage désigné par bits de parité, consiste à ajouter à la suite de chaque mot d'information, un nouveau bit, de telle sorte que le nombre total de chiffres 1 soit de parité fixé (paire ou impaire). Il s'agit d'un code systématique dont la clé de contrôle ne possède qu'un seul bit [6]. Le bit de parité est calculé à la base du nombre total de 1 soit toujours pair, à titre

Mot d'information : 100 1101 → Mot de code : **0**1001101

Mot d'information : 110 0111 → Mot de code : **1**1100111

1.4.1.3 Le code de Hamming

C'est un code correcteur d'erreur linéaire, il détecte et corrige automatiquement l'erreur, mais il corrige une seule erreur pas deux, valable pour des taux d'erreurs faibles sur une lettre de message, c'est un code parfait, ce qui signifie que pour une longueur de code donnée, il n'existe pas d'autre code plus compact ayant la même capacité de correction. En ce sens, son rendement est maximal, il est de paramètres :

- n: c'est la longueur totale de message.
- k: c'est les bits de message à transmettre.
- n -k: c'est les bites de contrôle.

1.4.1.4 Les codes cycliques

Ce sont des codes linéaires stables par permutation circulaire des coordonnées, qui ont une forte structure mathématique, c'est à dire, par les espaces vectoriels de polynômes, à savoir que les coordonnées des mots deviennent les coefficients des polynômes [3], et qui ont la propriété de stabilité par permutation circulaire des mots.

$T : F_2^n \rightarrow F_2^n$

$(X_1, X_2, \ldots\ldots X_n) \rightarrow (X_2, \ldots\ldots X_n, X_1)$

On identifie F^n à l'algèbre $F_2[X]/(X^n-1)$ par $(x1, x2, \ldots\ldots, x) \rightarrow x_1X^{n-1} + \ldots + x_{n-1}X + x$

Ce sont des codes polynomiaux dont le polynôme générateur $g(x)$ divise (x^n+1) où n est la longueur du code. Cette particularité permet la construction immédiate d'une matrice de contrôle caractéristique de ce type de code et une simplification de la méthode de correction automatique.

Pour la détermination des codes cycliques de longueur n, la connaissance des diviseurs (x^n+1) est essentielle [6].

1.4.1.5 Les codes BCH

Bose-Chaudhuri-Hocqenghem, ou par abréviation, les codes **BCH**, ce sont des codes cycliques, qui ont été découverts à la fin des années 50, l'acronyme est composé des lettres des noms de ses inventeurs, Raj Bose, D. K. Ray-Chaudhuri et Alexis Hocquenghem. Ces codes sont construits sur un alphabet composé d'un grand ensemble de symboles basés sur les propriétés des corps finis, qui définissent une méthode systématique pour construire des codes cycliques capables de corriger un nombre « *t* » d'erreurs arbitrairement fixés dans un bloc de « N » éléments binaires.

Les performances de décodage de ces codes sont directement liées à la distance de Hamming du code en bloc utilisé, plus cette distance est grande, plus on aura une meilleure performance de décodage. Ces codes sont utilisés dans des applications pour les communications satellitaires [3].

1.4.1.6 Les codes Reed-Solomon

C'est l'objectif de notre travail élaboré, Ces codes appartiennent à la classe des codes correcteurs d'erreurs cycliques non binaires, formés de *n* symboles, d'où,« $n = q - 1$ » et « $q = 2^s$ », chaque symbole appartenant à GF(q) qui est le corps de Galois (Galois Field) à *q* éléments, *s* représente le nombre de bits par symbole, pour une distance minimale « *d* » et une dimension « $k = n - d + 1$ ». A cet effet, on va voir ces paramètres, à savoir :[*n, k, n - k* + 1]. Le nombre « t » est égal à *(n-k) /2* représente le nombre de symboles d'erreurs, ce code sera capable.

C'est l'objectif de notre travail élaboré, Ces codes appartiennent à la classe des codes correcteurs d'erreurs cycliques non binaires, formés de *n* symboles, d'où,« $n = q - 1$ » et « $q = 2^s$ », chaque symbole appartenant à GF(q) qui est le corps de Galois (Galois Field) à *q* éléments, *s* représente le nombre de bits par symbole, pour une distance minimale « *d* » et une dimension « $k = n - d + 1$ ». à cet effet, on va voir ces paramètres, à savoir :[*n, k, n - k* + 1]. Le nombre « t » est égal à *(n-k) /2* représente le nombre de symboles d'erreurs, ce code sera capable.

1.4.1.7 Les codes LDPC

Les codes LDPC en abréviation ou Lowe Density Parity Check ont été découverts par Gallager dans les années soixante [3]. Il a proposé une méthode générale pour construire des codes LDPC pseudo aléatoires, les bons codes LDPC sont générés par ordinateur (en particulier les codes longs) [9]. Ces codes demandent une grande complexité calculatoire qui n'était pas disponible au moment de leur découverte.

Ils sont restés discrets jusqu'au1996. MacKay, travaillant sur les Turbo codes à ce moment-là, qui a donné une deuxième naissance aux codes LDPC MacKay (1999), cela fut possible du fait de l'avancée de la technologie en électronique qui a permis de pouvoir implémenter les algorithmes de décodage de ces codes [3].

C'est une matrice de parité creuse H (le poids de chaque ligne vaut quelques unités, alors que la longueur peut atteindre plusieurs milliers). La performance de l'algorithme de décodage tient essentiellement à la faible densité de la matrice [1], qui contient très peu de 1 (< 3%), représentée sous la forme d'un graphe de Tanner, comme indiqué dans le graphe en **figure 1.10**, une matrice quelconque (non creuse), composé de N nœuds de variable et de N -K nœuds de parités. Un nœud de variable *i* est connecté à un nœud de parité j si H (i; j) = 1 [4].

Matrice quelconque Graphe correspondant :

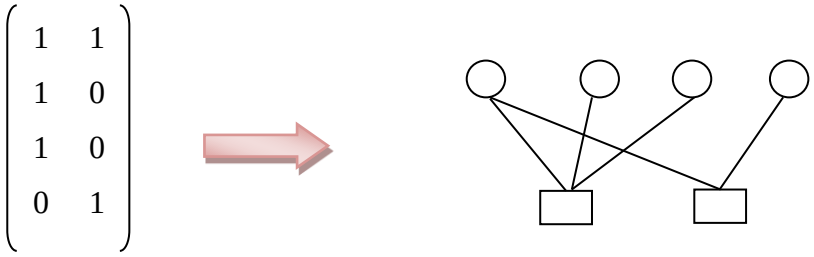


Figure 1.10: *Représentation d'une matrice de parité H d'un code LDPC et de son graphe de Tanner[4].*

1.4.2 Les codes convolutifs

Dans les années cinquante, Elias a introduit ce type des codes, qu'apparus plus tard, ils offrent des performances égales, supérieures dans beaucoup d'application pratiques aux codes en blocs, ce sont des codes plus utilisés dans les systèmes de télécommunications fixes et mobiles, plus faciles à implémenter et utilisables en temps réel [7].

Il se diffère d'un code bloc par le fait que chaque bloc de « *n* » éléments en sortie ne dépend pas seulement des k entrées à un instant donné (**figure 1.11**) [10]

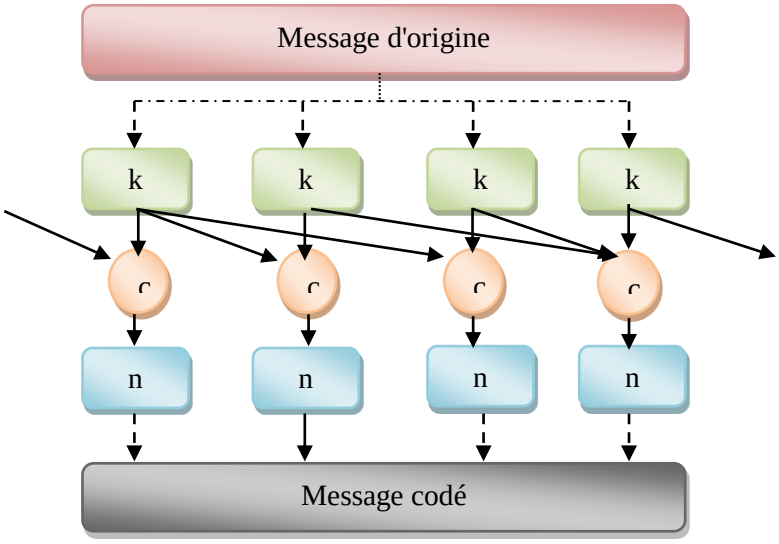


***Figure1.11** : Codage convolutif* [10].

La structure des codes convolutifs est riche et permet plusieurs approches.

1.4.2.1 Principe de fonctionnement

Le codeur d'un code convolutif admet en entrée une séquence d'information constituée de blocs de « K » bits, et qui génère une séquence de blocs de « N « bits. En codage convolutif un bloc de sortie n'est pas relié qu'à un seul bloc d'entrée mais à plusieurs [7].

Il est défini à l'aide de trois paramètres (n ; k ; m), à savoir, « n » : le nombre de sorties, « k » : le nombre d'entrées et « m » : la taille de la mémoire. Dans chaque unité de temps, le codeur lit k bits d'information et produit n bits codés (on à n > k) [1], ce codeur est composé de (m+1) registres de K bits pour stocker les m entrées précédentes et l'entrée actuelle, alors, cette quantité, (m+1), est appelé la longueur de contrainte du code. Le rapport $\frac{k}{n}$ est appelé le rendement du code [3]. Les paramètres K et N prennent en des valeurs inférieures à celles prises par des codes en blocs équivalents, et pour une probabilité d'erreur de décodage soit très faible, on choisira une taille de mémoire suffisamment grande [7].

1.5 Etude comparatifs des déférant codes

A la lanterne de ce qui précède, on constate qu'il existe plusieurs types de code dont chacun d'eux possède des caractéristiques spécifiques et une différence par rapport à l'autre, tous comme on a expliqué auparavant. Fragi et al.[11], ont analysé les performances de certains codes comparaisons comme : le code Reed-Solomon (RS) et les blocs de circuits turbo et par des simulations explicites par le chercheur, ils ont montré que les codes turbo offrent un BER bien plus petit que celui du code RS et que ce dernier présente une capacité de correction meilleure que celle des codes CRC qui ne font que détecter l'erreur sans la corriger. Aussi, dans un travail similaire Diop et al. [12],

Ont montré que les meilleurs rapports PSNR sont obtenus par le codage turbo à base du code convolutif et qu'il présente un taux du PSNR plus important que celui de Reed-Solomon. On va présenter ci-dessous un tableau récapitulatif des caractéristiques de quelques codes afin d'avoir une idée plus claire sur les différences existantes entre les différents codes.

Les codes	***Domains d'application***	***Avantages***	***Inconvenient***
Code de répétition	- En télécommunication pour reconnaître une modulation du signal.	- Protéger simultanément plusieurs bits d'information - Permet de corriger n/2 erreurs.	- Rendement faible
Code de parité	- Dans les transmissions séries	- Détection d'une seule erreur .	- Ne permet pas de corriger l'erreur .
Code de Hamming	- Dans les transmissions signal et En télécommunications	- Permet de détecter et corriger automatiquement les erreurs -Un rendement important .	- permet de corrigé une seul erreur .
Code cyclique	- Une représentation polynomiale des bits à transmettre .	- Détecter toutes les erreurs simple et doubles . - Plus performant que les simples checksums , surtout pour les paquets rafales d'erreurs .	- Moins coûteux en taille.
Code BSH	- En communication satellitaires .	- Permet de corriger un nombre t d'erreur arbitrairement fixés dans un bloc de N éléments binaires . - Une grande capacité de correction d'erreur. - Bon rapport signal sur bruit -Faible complexité	- Rendement faible .
Code Reed-Solomon	- Dans les communications mobile et réseaux sans fils , Dans les communications satellitaires , dans la télévision numérique et la radio diffusion numérique DVB Dans les modems ADSL et VDSL Dans la sauvegarde de données (sauvegarde magnétique , optique ...etc) . -Les codes concaténés . CD et DVD.	- Efficace dans la correction d'erreurs par paquet.	- Une grande complexité de calcules.
Code LDPC	- Application temps réel comme le stockage magnétique . - Les réseau locaux sans fil avec un débit élevé (WLAN) . - Systèmes CDMA et OFDM .	- Ne nécessite pas d'entrelacereur pour réaliser une bonne performance d'erreur. - Une meilleure performance par trame . -Marge d'erreur se produit à un niveau de BER plus faible . - Le décodage n'est pas serial et peut être réalisé par un processus parallèle .	- Une grande complexité de calcules.

Tableau 1.1 : *Comparaison entre quelques types de codes.*

1.6 Conclusion

On a présenté dans ce chapitre, le schéma général d'une transmission, puis on a rappelé quelques définitions de quelques types de codes comme le code binaire et le code linéaire avec des généralités sur leurs techniques de décodage, les différents étages composant une transmission numérique lorsque l'étage du codage fait une phase primordiale et déterminante en face des différentes perturbations rencontrées pendant la transmission, qui peuvent dégrader amplement la qualité du signal par la création d'éventuelles erreurs au niveau des informations envoyées., dont l'importance des codes correcteurs des erreurs pour récupérer les pertes des données et restituer le contenu du signal transmis. Pour cet effet, nous avons consacré la grande partie de ce chapitre à introduire les codes correcteurs d'erreurs en décrivant leurs différents types et les caractéristiques de chaque type de code, on a vu aussi, les relations qui existent entre les codes correcteurs d'erreurs et la technique de correction adoptée par chaque type de code, les mécanismes qui régissaient le codage et le décodage d'informations, mais ce n'était qu'un bref aperçu du monde des codes correcteurs d'erreurs.

A l'issu, on a conclue dans ce chapitre, les caractéristiques de quelques types de code dans un tableau récapitulatif qui permet d'avoir une idée générale sur le domaine d'application et les performances positives et négatives de chaque type de codage, avec les avantages et les inconvénients de chaque code.

CHAPITRE 2 :
LE CODE REED-SOLOMON

2.1 Introduction

Dans ce chapitre, on va étudier les caractéristiques et les principes de fonctionnement du code de Reed-Solomon, qui est largement appliqué suite à sa forte importante envers la correction des erreurs, puis, on va présenter aussi des exemples explicites sur la détection et la correction des erreurs.

A cet effet, on va voir dans ce chapitre ce code précité, d'où on va définir l'opération de codage et de décodage pas à pas en détaillant le jeu de calcul et l'équation formant le système de codage/décodage RS « Reed-Solomon ».

2.2 Les codes de Reed-Solomon (RS)

Les codes de Reed-Solomon (RS) sont développés par ***Irving Reed*** et ***Gustave Solomon*** en 1958 [7]. Ces codes sont certainement les codes par blocs les plus utilisés pour la correction d'erreurs dans les CD, les DVD et la plupart des supports de données numériques, Ils sont très utilisés car ils sont puissants du point de vue de la capacité de protection [18]. Il s'agit de codes adaptés à la correction des paquets d'erreurs [10].

Les messages sont divisés en blocs et on a ajouté des informations redondantes à chaque bloc permettant ainsi de diminuer la possibilité de la retransmission. La longueur du bloc dépend de la capacité du codeur. Avant de procéder aux mécanismes de codage/ décodage, on précède par introduire quelques notions élémentaires.

2.3 Théorie de Reed-Solomon

Reed-Solomon est un code de bloc peut être spécifié comme RS (n, k) et comme le montre la **Figure 2.1**. La variable « n » est la taille du mot de code avec l'unité des symboles, « k » est le nombre de + symboles de données et 2t est le nombre de symboles de parité. Chaque symbole contient « m » nombre de bits.

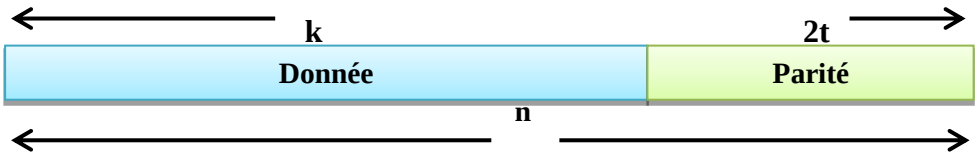


Figure 2.1: Structure d'un mot de passé RS

Les codes RS sont généralement représentés sous forme de RS (n, k), avec des symboles à m bits, où :

- Longueur du bloc: n
- Nombre de symboles du message d'origine: k
- Nombre de chiffres de parité: n - k = 2t
- Distance minimale: d = 2t + 1.

La relation entre la taille du symbole, m, et la taille du mot de code n, est donnée par un canal de communication, de transfert de données protégé par Reed-Solomon, comme illustré dans la **Figure 2.2.**

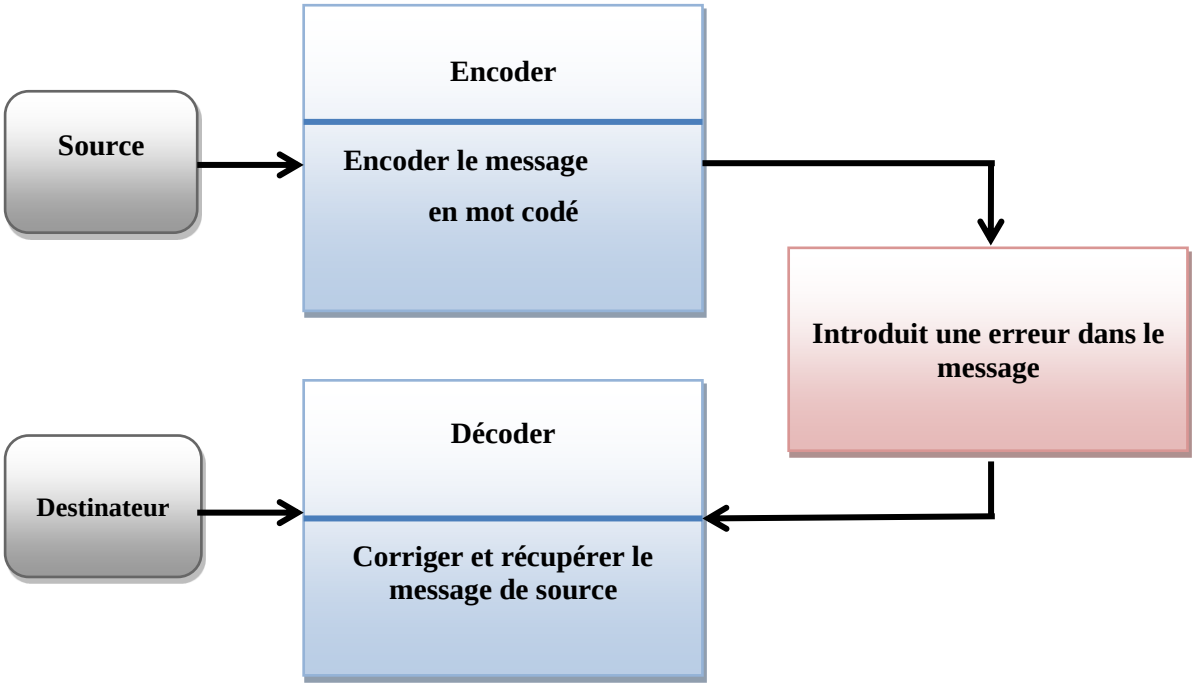


Figure 2.2 : Canal de transfert de données Reed-

Le codeur RS fourni à l'extrémité de l'émetteur code le message d'entrée en mot codés et le transmet via le canal. Le bruit et d'autres perturbations dans le canal peuvent perturber et corrompre le mot codé. Ce mot codé corrompu arrive à l'extrémité du récepteur (décodeur), d'où, il est vérifié et le message corrigé est transmis au récepteur. Si l'erreur induite par le canal est supérieure à la capacité de correction d'erreur du décodeur, un échec de décodage peut se produire.

On dit que des échecs de décodage se sont produits si un mot codé est passé inchangé, un échec de décodage, par contre, conduira à un mauvais message étant donné à la sortie [1].

Les codes de Reed-Solomon sont des codes correcteurs d'erreurs utilisés dans tous les domaines requérant des données fiables. Typiquement, dans les communications spatiales, télévision numérique et stockage de données. Les codes de Reed-Solomon permettent de corriger des erreurs et des effacements grâce à des symboles de contrôle ajoutés après l'information **[16].**Aussi, la nature de ces codes a permis d'aboutir à des algorithmes de codage et de décodage très performants y'a compris dans le cas de codes en blocs de grande taille. Nous allons par la suite entamer l'étude sur les codes RS.

2.4 Les champs de Galois

On définit le champ fini, un champ avec un ordre de champ fini qui veut dire un nombre des éléments), appelé aussi le corps de Galois. L'ordre d'un corps fini est toujours par un premier ou une puissance d'un premier. Pour chaque première puissance, il existe un corps fini GF (p^m). On dit infini s'il est constitué d'un nombre infini d'éléments, à titre exemple, l'ensemble de nombres réels, et complexes, et quand on consiste un nombre fini d'éléments.

GF(p^m) : un champ d'extension du champ sol GF(p), où m est un entier positif. Pour p = 2, GF (2^m) est un champ d'extension du champ sol GF(2) de deux éléments (0, l). GF (2^m) est un espace vectoriel de dimension m sur GF(2) et donc représenté en utilisant une base de m vecteurs linéairement indépendants. Le champ fini GF (2^m) contient (2^m - 1) éléments non nuls, tous les champs finis contiennent un élément nul et un élément α, appelé générateur ou élément primitif, de sorte que chaque élément non nul du champ est exprimé comme une puissance de cet élément. L'existence de cet élément primitif (d'ordre 2^m -1) est affirmée par le fait que les éléments non nuls de GF (2^m) forment un groupe cyclique.

Les codeurs et décodeurs pour coder de blocs linéaires sur GF (2^m), tels que les codes Reed-Solomon, nécessitent des opérations arithmétiques en GF(2^m), de plus, les décodeurs pour certains codes sur GF (2), tels que les codes BCH, nécessitent des calculs dans les champs d'extension GF (2^m). Dans GF (2^m), l'addition et la soustraction sont simplement des exclusifs au niveau du bit. La multiplication est effectuée par plusieurs approches, y compris les bits série, bit parallèle (combinatoire) et logiciel. La division nécessite l'inverse du diviseur, qui est calculé en matériel à l'aide de plusieurs méthodes, notamment l'algorithme d'Euclid, les tables de recherche, l'exponentiation et les représentations de sous-champs. À l'exception de la division, les circuits combinatoires pour l'arithmétique de champ de Galois sont simples, et la plupart des algorithmes de décodage peuvent être modifiés de manière nécessite que quelques divisions, de sorte que les méthodes rapides de division ne sont pas essentielles.

2.4.1 Propriétés des Champs Galois

Dans les champs GF (2^m), il y a toujours un élément primitif α, tel on peut exprimer chaque élément de GF (2^m) sauf zéro comme une puissance de **α** [26]. On peut générer chaque champ GF (2^m) en utilisant un polynôme primitif sur GF(2), dont l'arithmétique effectuée dans le champ GF (2^m) est modulo ce polynôme primitif.

- Si α est un élément primitif de GF (2^m), il se conjugue α^{2m}, aussi des éléments primitifs de GF(2^m).

- Si α est un élément d'ordre n dans GF(2^m), tous ses conjugués ont le même ordre n.
- Si α, un élément de GF(2^m), est une racine d'un polynôme f (x) sur GF(2), alors tous les conjugués distincts également, éléments de GF(2^m), qui sont des racines de f (x).
- Les éléments 2^m - 1 non nuls de GF(2^m) forment toutes les racines de x^{2m} - 1 - 1 = 0. Les éléments de GF (2^m) forment toutes les racines de x^{2m} - x = 0.
- Soit α dans GF(2^m), φ (x) est le polynôme minimal de α sur GF (2), si φ (x) est le polynôme de plus petit degré sur GF(2) tel que φ (α)= 0.
- Chaque élément α de GF (2^m) a un polynôme minimal unique sur GF(2).
- Le polynôme minimal φ (x) d'un élément de α qui est un champ irréductible.
- Un polynôme irréductible f(x) sur GF(2) est le polynôme minimal d'un élément α dans GF (p^m) si f (α) = 0.
- Soit φ (x) le polynôme minimal de α, si α est la racine de f (x), alors f (x) est divisible par α(X).
- Le polynôme minimal φ (x) d'un élément α dans GF (2^m) divise x^{2m} - x.
- Le polynôme minimal φ (x), également irréductible, d'un élément α dans GF (p^m) est :

$$\varphi(x) = \prod_{i=0}^{e-1}(x + \alpha^{2m}) \quad \textbf{(2.1)}$$

- Soit e le degré d'un polynôme minimal φ (x) d'un élément α dans GF(p^m), alors e est le plus petit entier tel que $\alpha^{pe} = \beta$, et e ≤ m, e divise m.
- Toute opération sur des éléments du terrain conservera la propriété de fermeture. Chaque élément a un inverse additif et multiplicatif (sauf pour «0» qui n'a pas d'inverse multiplicatif).

2.4.2 Construction des Champs de Galois

Un champ de Galois GF (2^m) avec élément primitif α est généralement représenté par (0, 1, α, α 2, ……. α^{2^k-2}). L'exemple le plus simple d'un champ fini est le champ binaire constitué des éléments (0, 1). Traditionnellement appelé GF $(2)^2$, les opérations dans ce champ sont définies comme l'addition d'entiers et la multiplication réduite modulo 2. Les champs les plus grands peuvent être créés en étendant GF(2) dans un espace vectoriel conduisant à des champs finis de taille 2^m. Ce sont des simples extensions du champ de base GF (2) sur m dimensions. Le champ GF (2^m) est ainsi défini comme un champ de 2^m éléments dont chacun est un m-tuple binaire. En utilisant cette définition, m bits de données binaires qui peuvent être regroupés et référencés comme un élément de GF (2^m). Cela permet à son tour d'appliquer les opérations mathématiques associées du champ pour coder et décoder des données **[10].**Soit le polynôme primitif φ (x), de degré m sur GF (2^m). Maintenant, tout champ est donné par le $i^{ème}$ élément :

$$a_i(\alpha) = a_{i0} + a_{i1}\alpha + a_{i2}\alpha + \ldots\ldots\ldots + a_{im}\alpha^{m-1} \quad \textbf{(2.2)}$$

Par conséquent, tous les éléments de ce champ peuvent être générés comme des puissances de α. Ceci est la représentation polynomiale des éléments du champ, qui suppose également que le coefficient principal de φ (x) est égal à 1, comme indiqué dans la figure 2.1, qui montre le champ fini généré par le polynôme primitif $1 +\alpha + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^8$ représenté par GF (2^8) ou GF (256).

Puissance	Polynôme	4 tuples binaire	Décimale	Indexe
0	0	0 0 0 0	0	-1
1	1	1 0 0 0	1	0
α^1	A	0 1 0 0	2	1
α^2	α^2	0 0 1 0	4	2
α^3	α^3	0 0 0 1	8	3
α^4	1 A	1 1 0 0	3	4
α^5	A α^2	0 1 1 0	6	5
α^6	α^2 α^3	0 0 1 1	12	6
α^7	1 A α^3	1 1 0 1	11	7
α^8	1 α^2	1 0 1 0	5	8
α^9	A α^3	0 1 0 1	10	9
α^{10}	1 A α^2	1 1 1 0	7	10
α^{11}	A α^2 3	0 1 1 1	14	11
α^{12}	1 A α^2 α^3	1 1 1 1	15	12
α^{13}	1 α^2 α^3	1 0 1 1	13	13
α^{14}	1 α^3	1 0 0 1	9	14

Tableau 2.1 : éléments de GF (2^8) [16].

A noter que, le polynôme primitif est de degré 4, la valeur numérique de α est considérée comme purement arbitraire. En utilisant la propriété d'irréductibilité du polynôme φ (x), on trouve que cette construction produit bien un champ.

Le polynôme primitif est utilisé dans un algorithme itératif simple pour générer tous les éléments du champ. Par conséquent, différents polynômes généreront des champs différents. S'il existe de nombreux choix pour les polynômes irréductibles [29], les champs construits sont tous isomorphes.

2.4.3 Champ de Galois Arithmétique :

L'arithmétique de champ de Galois (CGA) est très intéressante à plusieurs égards. Toutes les opérations qui entraînent généralement un débordement ou un sous-dépassement en mathématiques traditionnelles sont mappées sur une valeur à l'intérieur du champ en raison de l'arithmétique modulo suivie dans CGA, les problèmes d'arrondi sont donc automatiquement éliminés.

CG facilite généralement la représentation de tous les éléments avec un mot binaire de longueur finie. Dans GF (2^m), tous les opérandes sont de m bits, où m est toujours un nombre inférieur à la largeur de bus conventionnelle de 32 ou 64.

Ceci à son tour introduit une quantité énorme de parallélisme dans les opérations CGA. De plus, nous supposons que «m » est un multiple de 8, ou une puissance de 2, à raison de sa commodité inhérente au parallélisme des sous-mots [30].

Pour étudier le CGA, une introduction aux concepts mathématiques de trace et de base duale est nécessaire [31], [32].

Définition 1: La trace d'un élément β appartenant à GF (2^m) est définie comme :

$$Tr(\beta) = \sum_{k-0}^{m-1} (\beta)^{2k} \quad \textbf{(2.3)}$$

Définition 2 : Une base $\{\mu_j\}$ dans GF (2^m) est un ensemble de m éléments linéairement indépendants dans GF (2^m), d'où : $0 \leq j \leq m-1$

Définition 3 : Deux bases $\{\mu_i\}$ et $\{\lambda_i\}$ sont duales l'une de l'autre si :

$T_r(\mu_i\lambda_i\} = 1$ si $j = k$;

$T_r(\mu_i\lambda_i\} = 0$ si $j \neq k$;

Les éléments de GF (2^m) sont généralement exprimés en puissances de l'élément α primitif, d'où α est défini comme la racine du polynôme primitif :

$g(x) = x^m + g_{m-1} x^{m-1} + g_{m-2} x^{m-2} + \ldots\ldots\ldots\ldots + g_1 x + 1$ **(2.4)**

Ou $f_i \in \{0,1\}$. Chaque élément Z de GF (2^m) peut également être écrit dans [31], [32], [33].

La base standard comme $Z = a_{m-1} \alpha^{m-1} + a_{m-2} \alpha^{m-2} + \ldots\ldots\ldots + a_2 \alpha^2 + a_1 \alpha^1 + a_0$

La base normale comme $Z = a_{m-1} \alpha^{2m-1} + \ldots\ldots + a_2 \alpha^2 + a_1 \alpha^1 + a_0 \alpha$

La double base λ_k comme $Z = \sum_{k=0}^{m-1} Zk\,\lambda k = \sum_{k=0}^{m-1} Tr\,(Z\mu k)\lambda k$

Où : a_i= GF(2) et $Z_k = Tr(z\mu_k)$ est le $k^{ième}$ coefficient de la base double.

- **Addition / Soustraction**

En général, le champ GF (2^m) représente un ensemble d'entiers de zéro à $2^m - 1$. L'addition et la soustraction d'éléments de GF (2^m) sont de simples opérations XOR des deux opérandes. Chacun des éléments du GF est d'abord représenté comme un polynôme correspondant. L'opération d'addition ou de soustraction est alors représentée par l'opération XOR du coefficient de polynômes correspondants. Cependant, comme les opérations les plus complexes sont largement utilisées dans les algorithmes de codage et de décodage RS, le développement de leurs structures matérielles a subi une attention considérable.

A noter que **CGA** ne fait pas la distinction entre les opérations d'addition et de soustraction; les deux sont considérés comme des opérations XOR qui suivent l'arithmétique modulo, le résultat est toujours évalué à une valeur dans le champ.

- **Multiplication**

L'opération de multiplication sur le champ de Galois est une opération plus complexe que l'opération d'addition / soustraction. Il est basé sur l'arithmétique modulo, mais diverses approches avec des complexités variables ont été suggérées et adoptées jusqu'à ce jour. La majorité de ces approches sont décrites et évaluées dans cette sous-section.

Approche 1

$A = \alpha_0 + a_1\ \alpha + \ldots\ldots + a_{m-1}\ \alpha^{m-1}$ et $B = b_0 + b_1\alpha + \ldots + b_{m-1} + \alpha^{m-1}$ les deux éléments de GF(2^m), puis : $A+B=C= c_0+c_1\alpha+\ldots\ldots+c_{m-1}\alpha^{m-1}$ **(2.5)**

Il s'agit de l'opération d'addition normale, réalisée en suivant la méthodologie décrite ci-dessus. La multiplication sur GF (2^m) est définie par :

a(x) x b(x) = a(x)b(x) mod f(x) **(2.6)**

La première étape de ce calcul est la construction des produits partiels

$P_i(x) = a(x)b_ix^i \qquad (0 \le i \le m)$

Ensuite, tous les produits partiels obtenus $P_i(x)$ doit être ajouté modulo 2. Un résultat partiel $C_p(x)$ est :

$$Cp(x) = \sum Pi(x) mod 2 \quad \textbf{(2.7)}$$

$C(x)= c_0+c_1x+\ldots..+c_{2m-1}x^{2m-1}$ **(2.8)**

Ce résultat partiel n'est cependant pas un élément du champ GF(2). D'où le terme d'ordre le plus élevé x^h de $C_p(x)$ doit être réduit en utilisant la propriété suivante du polynôme irréductible :

$x= f_0 + f_1x+\ldots\ldots\ldots\ldots f_{m-1}x^{m-1}$ **(2.9)**

La substitution commence par le terme d'ordre $c_{2m-1}x^{2m-1}$ le plus élevé et produit un résultat partiel C_p d'un degré de moins que C_p à partir de (3,9). (3.10) est utilisé :

$C_p(x) = [c+c_1x+\ldots\ldots\ldots+c_{2m-2}x^{2m-2}] + [f_0x^{m-2}+ f_1xx^{m-2}+\ldots\ldots+f_{m-1}x^{m-1}x^{m-2}]$ **(2.10)**

Cette opération de décalage se poursuit jusqu'à ce que C_p (x) soit de degré m-1, il devienne un élément de GF (2^m). Le résultat d'une multiplication GF(2) est un mot de (2^{m-1}) bit, qui représente un polynôme de degré (2^{m-2}) sous forme étendue.

Pour m = 8, le produit est représenté comme suit: Considérons deux opérandes dans le GF (28),

$Op1= a_7\alpha^7+ a_6\alpha^6 + a_5\alpha^5\ a_4\alpha^4 + a_3\alpha^3+ a_2\alpha^2+ a_1\alpha^1+ a_0$ **(2.11)**

$Op2= b7\alpha7+ b6\alpha6 + b5\alpha5\ b4\alpha4 + b3\alpha3+ b2\alpha2+ b1\alpha1+ n0$ **(2.12)**

Le produit peut être représenté comme suit:

$Pr\ od\ (a) = Op1\ X\ Op2 = A_{14}\alpha^{14}+ A_{13}\alpha^{13}+ A_{12}\alpha^{12}+ A_{11}\alpha^{11}+ A_{10}\alpha^{10}+ A_9\alpha^9+ A_8\alpha^8+ A_7\alpha^7+ A_6\alpha^6+ A_5\alpha^5+ A_4\alpha^4+ A_3\alpha^3+ A_2\alpha^2+ A_1\alpha^1+A_0$ **(2.13)**

Le produit est un mot de 15 bits, contenant les coefficients du polynôme de forme étendue Prod(a). Ce mot contient les coefficients du polynôme de forme étendue Prod (a).

Le polynôme de forme étendue est réduit modulo f (x). Cela équivaut à calculer le reste à partir du polynôme de forme étendue divisé par f (x). Le reste dans ce cas est un polynôme de degré 7 à 8 coefficients binaires, représente un élément valide de GF (2^8).

Approche 2

Lorsque le champ est restreint à un petit nombre d'éléments finis, disons GF (2^k), où k≤ 16, alors l'approche suggérée dans [33], basée sur des tables de logarithmes adoptées.

Cette approche utilise deux tables :gflog et gfilog.

Pour le GF (2^m - 1), le tableau gflog est constitué d'indices de 1 à 2^m - 1, et met en correspondance ces indices avec leur logarithme correspondant dans le champ de Galois. Le tableau gfilog, est défini pour les indices 0 à 2^m -2, met en correspondance ces indices avec leur logarithme inverse respectif dans le champ de Galois.

Une opération de multiplication comprend désormais les trois étapes suivantes :

- Une opération de recherche dans la table gflog pour obtenir les valeurs du journal des opérandes d'entrée ;
- Les valeurs log sont ajoutées pour obtenir un résultat intermédiaire ;
- Une autre recherche de table, à partir de la table gfilog, donne le produit des deux opérandes.

Comme on a mentionné précédemment, CGA évite les problèmes de débordement et d'arrondi, d'où le résultat des opérations de journalisation est contrairement aux logarithmes réguliers qui donnent un entier pour tout élément différent de zéro dans le champ, conduisant à des opérations de multiplication précises.

Une implémentation efficace de ce schéma nécessite une conditionnelle, trois tables de recherche (deux pour les recherches de logarithme et une pour la recherche de log inverse), un additionneur et un circuit modulo [33].

Approche 3

Le travail [30] propose une méthode alternative pour la multiplication, une fois basée sur la recherche de table, l'opération de multiplication peut être effectuée sur des opérandes comme une simple opération modulo d'addition, après les avoir convertis « forme multiplicative » de l'opérande. Les trois étapes impliquées dans le calcul sont résumées ci-dessous.

- Les opérandes (qui sont à l'origine sous forme additive) sont convertis en forme multiplicative à l'aide de la table add&mul ;
- L'addition modulo est effectuée sur les opérandes convertis ;
- La valeur résultante est ensuite reconvertie sous forme additive à l'aide de la table mul/add.

La figure 3.2 indiquée au-dessous, illustre le fonctionnement de cette approche pour calculer c = a × b, qui a été adoptée à partir de [30].

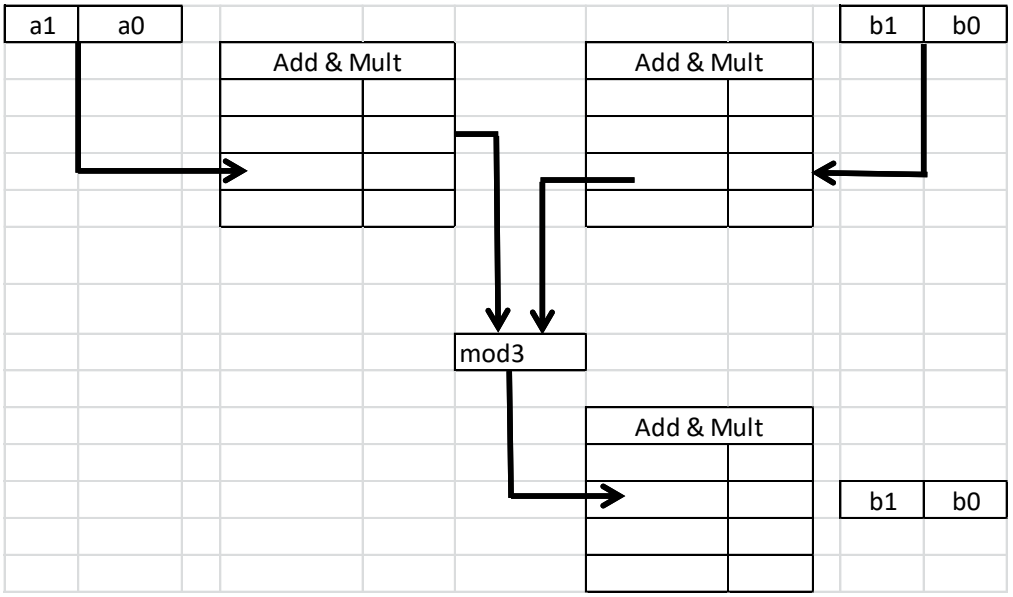


Figure 2.3 opération de multiplication

Un algorithme de division et de multiplication en série de bits a été présenté par Hasan et Bhargava en 1992 [34]. En utilisant les coordonnées des éléments de support, la division sur GF (q^m) est effectuée en résolvant un système de m équations linéaires sur GF(q) lorsque les éléments de champ sont représentés par des polynômes, en outre, il est montré que la division est effectuée avec un ordre inférieur de complexité de calcul en résolvant une équation de Wiener-Hopf de degré m, définie comme un système de m équations linéaires non homogènes, avec m inconnues [34].

2.4.4 Architectures de Multiplication Existantes

Les circuits de multiplication pour GF (2^m) peuvent être classés sur la base de leurs implémentations d'architecture: architecture à bits série, bit parallèle, hybride et série numérique. Les architectures série de bits [32] traitent un cycle bit / horloge, qui sont efficaces en zone et adaptées aux applications à faible surface. Les architectures bit-parallèles, traitant un cycle mot / horloge, qui sont idéales pour les applications à grande vitesse. Les architectures hybride et numérique série [35] ont récemment proposées pour compenser les compromis entre les architectures bit-serial et bit-parallèle.

L'architecture de multiplicateur hybride représente le champ GF(2^m) comme GF((2^n)^k). Il utilise l'arithmétique parallèle de bits dans le sous-champ GF(2^n) et le traitement en série pour l'arithmétique du champ d'extension. Aussi, le multiplicateur traite les données d'entrée à un débit de n bits par cycle d'horloge et donne des résultats de sortie à un rythme dans tous les k cycles d'horloge.

DIVISION

L'opération de division serait la plus consommatrice de temps et de surface de toutes les opérations sur les champs de Galois. L'approche générale pour effectuer la division sur GF (2^m) s'est de multiplier l'élément inverse d'un diviseur (β) par le dividende (γ). Le vrai problème pour effectuer cette opération de division est de trouver un algorithme de calcul efficace pour l'élément inverse multiplicatif du diviseur β avec le dividende γ :

$$\beta / \gamma = \beta \times \gamma^{-1} \quad \textbf{(2.14)}$$

Il faut connaitre deux algorithmes pour implémenter ce calcul qui sont :

1. Algorithme Euclids ;
2. Algorithme carré continu ;

2.4.4.1 Algorithme d'Euclid

L'algorithme euclidien détermine le plus grand diviseur commun (GDC) de deux entiers (sans nécessiter de factorisation). L'algorithme euclidien étendu est une version légèrement modifiée du même, calculer à la fois GDC et l'inverse multiplicatif d'un élément [28].

Étant donné deux entiers a et b, l'algorithme calcule d'abord le GDC des deux termes (a, b), ainsi que les entiers x et y tels que ax + by = gdc (a, b). Le calcul supplémentaire n'est pas très coûteux, d'autant que les étapes de l'algorithme d'Euclide traitent toujours des sommes de multiples de a et b. Le fonctionnement de la version étendue de l'algorithme d'Euclid est illustré dans laci-dessous **« figure 3.3 ».**

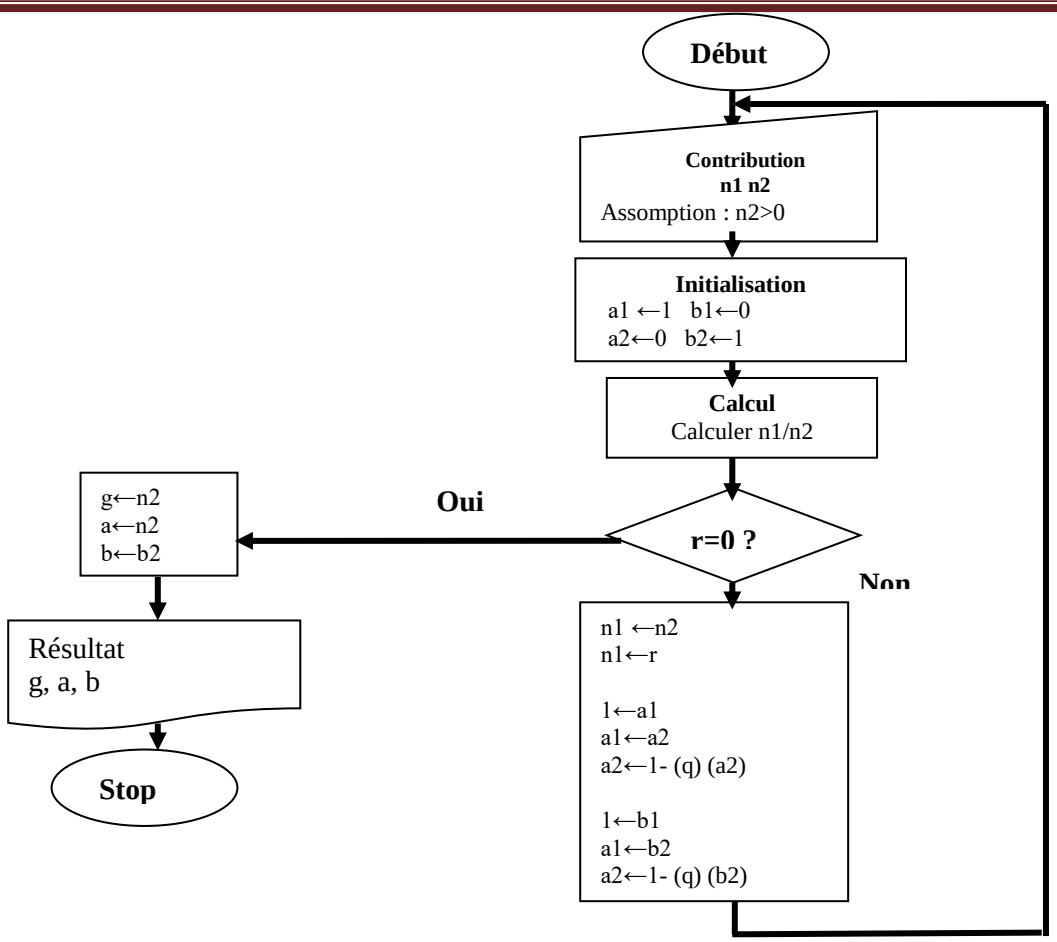


Figure 2.4 organigramme de l'algorithme d'euclidien étendu

L'équation ax + by = gdc (a, b) c'est la première importance dans la division de corps fini où a et b sont relativement premiers, car la valeur de x représentera maintenant l'inverse multiplicatif d'un modulo b.

2.4.4.2 Algorithme Carré Continu

Le fonctionnement de l'algorithme du carré continu est basé sur la propriété fondamentale de GF (2^m) que : $\gamma^{2m} - 1 = 1$ **(2.15)**

Équation multiplicatrice (3.15) avec γ^{-1} donne :

$\gamma^{-1} = \gamma^{2m-1} \times \gamma^{-1} = \gamma^{2m-2}$ **(2.16)**

Puisque :

$2^m - 2 = \sum_{i=1}^{m-1} 2^i$ **(2.17)**

γ^{-1} est calculé successivement par quadrillage et multiplication.

Il y a une autre approche alternative à la méthodologie ci-dessus consiste à suivre les mêmes étapes que dans l'approche 1 de multiplication, sauf qu'à l'étape 2, on soustraire les deux valeurs logarithmiques résultantes, au lieu de l'opération d'addition habituelle.

2.5 Propriétés de code RS « Reed-Solomon »

Les codes de Reed Solomon sont linéaires, qui font partie des codes **BCH** dont le codeur prend k symboles de donnée (chaque symbole contenant s bits) et calcule les informations de contrôle pour construire n symboles, ce qui donne n-k symboles de contrôle. Le décodeur peut corriger au maximum t symboles, où **2t=n-k.** Le diagramme ci-dessous montre une trame constituée par le codeur Reed-Solomon :

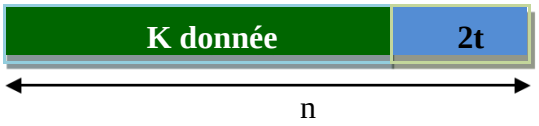


Figure 2.5: *mot-code de Reed-Solomon.*

La longueur maximale d'un code de Reed-Solomon est définie comme :

$$n = k+2t=2^{s}-1 \qquad \textbf{(2.18)}$$

Avec :

- k: nombre de symboles de données ;
- $2t$: nombre de symboles de contrôle ;
- s: nombre de bits par symbole.

La distance minimale d'un code Reed-Solomon est :

$$d_{min}= 2t+1 \qquad \textbf{(2.19)}$$

NB : Il utile à souligner que les codes de Reed-Solomon sont des codes non-binaires, représentés sur du champ de Galois $GF(2^m)$ et pas sur $GF(2)$ [16].

Exemple 2.7 :

On prend un code de Reed-Solomon(**15,9**), on l'utilise pour tous les autres exemples, dont le but est de savoir combien de bits utilisés pour chaque symbole et combien d'erreurs qu'on pourra corriger.

RS (n, k) =RS (15,9) n : s'indique la longueur totale d'un bloc de RS ; 15 symbole, et k indique la longueur du bloc d'information ; 9 symboles dans cet exemple.

La capacité de correction des erreurs du système est calculé comme suit:

$$2t=n-k=15-9=6$$

Donc :$t = \frac{n-k}{2} = 3$

- Dans cet exemple, ce code permettra de corriger 3 symboles ;
- Le nombre de bits s par symbole est comme suit:

$$n=2^{s}-1$$

$$S = \frac{\ln(n+1)}{\ln(2)} = \frac{\ln(16)}{\ln(2)}$$

Le nombre de bits utilisés pour coder les symboles est donc 4 bits, ce qui nous amène à utiliser un « champ de Galois » de GF**(2^4).**

2.6 Codage de RS « Reed-Solomon »

Dans le codes Reed-Solomon, le codage est effectué de la même façon que le codage à l'aide du **CRC**, la seule différence entre les deux, c'est que les codes de Reed-Solomon sont non-binaires (Reed-Solomon $GF(2^m)$), par contre les codes de CRC sont binaires, ($GF(2)$) **[16].**

2.6.1 Théorie du codage

L'équation clé qui définit le codage systématique de RS (n, k) est :

$$C(x) = i(x)\, x^{n-k} + [i(x)\, x^{n-k}] \bmod g(x) \qquad \textbf{(2.20)}$$

Avec:

- $C(x)$: polynôme du mot-code, degré$n-1$;
- $i(x)$: polynôme d'information, degré $k-1$;
- $[i(x)\, x^{n-k}] \bmod g(x)$: polynôme de contrôle, degré $n-k-1$;
- $g(x)$: polynôme générateur, degré $n-k$.

NB : il est utile à signaler que le codage systématique signifie que l'information est codée dans le degré élevé du mot de code, dont les symboles de contrôle sont introduits après les mots d'information **[16].**

2.6.1.1 Polynôme générateur

Soit m un entier positif supérieur à 2, et t un entier positif inférieur à 2^m-1 et un élément primitif de GF (2^m), on appelle g(x) le polynôme de degré le plus faible ayant les valeurs $\{\alpha, \alpha^2, \alpha^3 \ldots, \alpha^{2t}\}$ comme des racines de [16,21].

$$g(x) = \prod_{i=1}^{2t}(x - \alpha^i) \qquad \textbf{(2.21)}$$

$$g(x)=(x-\alpha^1)\,(x-\alpha^2)\,(x-\alpha^3)\ldots\ldots\ldots\ldots\,(x-\alpha^{2t}) \qquad \textbf{(2.22)}$$

Exemple 2.8: On calcule les coefficients du polynôme générateur pour Reed-Solomon RS (15,9):

On va calculer les coefficients du polynôme générateur pour déterminer des symboles de contrôle d'un code de RS (15,9) qui pourra corriger 3 erreurs, comme on a vu précédemment dans l'exemple **(2.7).**

La forme générale du polynôme générateur est donnée dans [16] :

$g(x)=(x-\alpha^1)\ (x-\alpha^2)\ (x-\alpha^3)\ldots\ldots\ldots\ldots\ (x-\alpha^{2t})$ **(2.23)**

En développant l'équation (2.13), on trouve :

$g(x)=(x-\alpha^1)\ (x-\alpha^2)\ (x-\alpha^3)\ldots\ldots\ldots\ldots\ (x-\alpha^{2t})$ **(2.24)**

$=(x^2+\alpha^5x+\alpha^3)\ (x^2+\alpha^7x+\alpha^7)(x^2+\alpha^9x+\alpha^{11})$

$=x^6+x^5(\alpha^{13}+\alpha^9)+x^4(\alpha^6+\alpha^7+\alpha^{11})+x^3(\alpha^3+1+\alpha^9)+x^2(\alpha^{12}+\alpha^{10}+\alpha^2)+x(\alpha^4+\alpha^{14})+\alpha^6$

$=x^6+\alpha^{10}x^5+\alpha^{14}x^4+\alpha^4x^3+\alpha^6x^2+\alpha^9x+\alpha^6$

➢ Simulation sur MATLAB

```
%      Polynôme générateur de
RS(15,9)  n=15;
     k=9;
     m=4;
     g=rspoly(n,k,m)
     g =
     6   9   6   4   14   10   0
```

On conclue, que le résultat des racines de Polynôme Générateur de RS (15,9) est comme suit:**(α^6 α^9 α^6 α^4 α^{14} α^{10} 1)**.

2.6.1.2 Mot de code

On va voir dans le polynôme C de degré n-l dont les coefficients appartiennent à GF(2), si {**α α^6, $\alpha^2\cdots\alpha^{2t}$**} sont des racines de C(x), donc, par conséquent, C(x) est aussi divisible par g(x) [21], le code Reed-Solomon RS (n, k) est codé comme un code binaire BCH, dont la seule différence est que les multiplications et les additions doivent réalisés dans GF(2^m).

Pour le codage systématique d'un polynôme de message m(x) avec k coefficients ayant le k symboles de message sur GF(2^m); le polynôme de vérification de parité m(x) c'est le reste du polynôme de messages décalés X^{n-k} m(x) divisé par le polynôme générateur *g(x)* [14], donc;

$\mu(x)= X^{n-k}\, m(x) \bmod g(x)$ **(2.25)**

Le mot de code est comme suit :

$C(x)= \mu(x) + X^{n-k}\, m(x)$ **(2.26)**

Exemple 2.9:

Considérant que le code linéaire cyclique RS défini dans GF (2^3) pour corriger 2t (t=2) erreurs, et les paramètres de ce code sont :

n= 2^p-1=2^3-1=7 et k=2^p-1-2t=8-1-4=3

Le polynôme générateur de ce code qui vérifie la formule :

$$G(x) = \prod_0^3 (x - \alpha^i) = x^4+\alpha^2x^3+\alpha^5x^2+\alpha^5x+\alpha^6 \quad \textbf{(2.27)}$$

On suppose qu'on veut coder le mot m(x)= $x^2+\alpha x+ \alpha^2$, alors, pour le codage à logique majoritaire, on doit calculer :

$\mu(x)= X^{n-k} m(x) \bmod g(x)$

$$\mu(x)= X^4(x^2+\alpha x+\alpha^2) \bmod (x^4+\alpha^2x^3+\alpha^5x^2+\alpha^5x+\alpha^6) \quad \textbf{(2.28)}$$

$$\begin{array}{l|l}
\alpha^6+\alpha x^5+\alpha^2 & x^4+\alpha^2 x^3+\alpha^5 x^2+\alpha^5 x+\alpha^6 \\
-x^6-\alpha^2x^5-\alpha^5x^4-\alpha^6x^2 & \alpha^2+\alpha^4x+\alpha^4 \\
\hline
\alpha^4x^5+\alpha^3x^4+\alpha^5x^3+\alpha^6 & \\
-\alpha^4x^5-\alpha^6x^4-\alpha^9x^3-\alpha^9x^2-\alpha^{10}x & \\
\hline
\alpha^4x^4+\alpha^3x^3+\alpha^0x^2+\alpha^3x & \\
-\alpha^4x^4-\alpha^6x^3-\alpha^9x^2-\alpha^9x^1-\alpha^{10} & \\
\hline
\alpha^4x^3+\alpha^6x^2+\alpha^5x+\alpha^6x^2 &
\end{array}$$

Donc: $\mu(x)=\alpha^3+\alpha^4 x^3+\alpha^5x+\alpha^6 x^2$

Le mot code est :$C(x)= \mu(x) + X^{n-k}m(x)$

$$C(x)= \alpha^3+\alpha^4 x^3+\alpha^5x+\alpha^6 x^2+x^2+\alpha x+\alpha^2$$

A l'issu, le mot de code est donc : **[α^3 α^4 α^5 α^6 1 α α^2]**

➢ Application sur MATLAB

```
% Le mot de code de Reed-Solomon(7,3)
n=7;
k=3;
m= [0 1 2];
c=rsenco(m,n,k,'power')
c =
   3   4   5   6   0   1   2
```

On conclue que le résultat des racines de mot de code RS(7,3)est comme suit :

(α^3 α^4 α^5 α^4 α^6 1 $\alpha\alpha^2$).

2.6.2 Décodage de Reed-Solomon « RS »

La base du décodeur RS est de détecter une séquence erronée avec peu de termes, sommée aux données reçues, qui donnera lieu à un mot-code valable, il y a plusieurs étapes nécessaires pour le décodage de ces codes :

- Le calcul du syndrome ;
- Le calcul des polynômes de localisation des erreurs et d'amplitudes ;
- Le calcul des racines, évaluations des deux polynômes, et sommation du polynôme constitué et du polynôme reçu pour reconstituer l'information de départ sans erreur [16].

Pour plus des détails, le diagramme ci-dessous (Figure 2.2) inique les étapes de décodage.

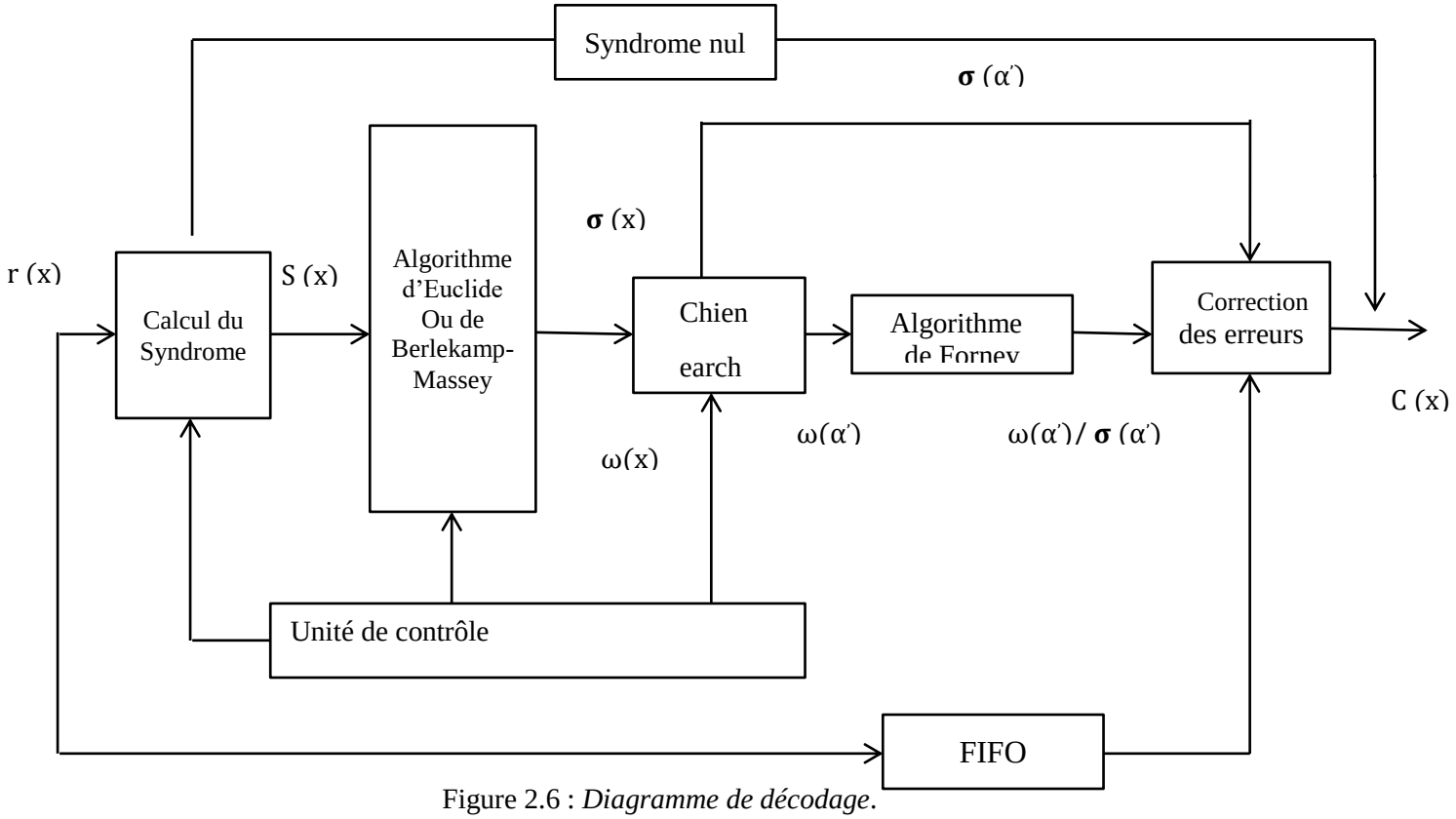


Figure 2.6 : *Diagramme de décodage*.

r(x) : Mot de code reçu ;

S(x) : Syndrome calculé ;

ω (x) : Polynôme d'amplitude des erreurs ;

ω(α'): Polynôme d'amplitude des erreurs, pour tous les éléments y compris dans GF (2^m) ;

σ (x) : Polynôme de localisation des erreurs ;

σ (α') Polynôme de localisation des erreurs, pour tous les éléments y compris dans *GF*(2^m) ;

σ' (α') : Dérivée du**σ (α')**.

$\frac{\omega\ (\alpha`)}{\sigma`(\alpha`)}$: c'est la division entre le Polynôme d'amplitude et ω(α')

c(x) : c'est le mot de code reconstitué.

2.6.2.1 Phases du décodage

On considère un code de Reed-Solomon *c(x)*correspond au code transmis, soit *r* (*x)* le code que l'on reçoit, le polynôme d'erreur introduit par le canal est défini comme suit :

$$e(x) = r(x) - c(x) = r(x) + c(x) \quad \textbf{(2.29)}$$

$$=e_0+e_1x+\ldots..e_{n-1}x^{n-1}$$

On suppose que le polynôme d'erreur contient des erreurs dans les positions :

$x^{j1}, x^{j2},\ldots\ldots, x^{jv}$

avec $0\leq j_1<j_2<\ldots\ldots<j_v\leq n-1,$

on peut alors définir le polynôme des erreurs comme suit:

$$e(x) = e_{j1}x^{j1} + e_{j2}x^{j2} + \ldots\ldots\ldots + e_{jv}x^{jv} \quad \textbf{(2.30)}$$

Avec :

$e_{j1}, e_{j2},\ldots..e_{jv}$: des valeurs d'amplitude des erreurs

$x^{j1},x^{j2}, \ldots x^{jv}$: c'est la position des erreurs

On peut calculer le polynôme du syndrome *S* (*x),* à partir du polynôme *r* (*x)* reçu, qui va nous indiquer la présence d'éventuelles erreurs, et si tous les coefficients du syndrome sont nuls, donc, les étapes du décodage n'ont pas lieu d'êtres, car le mot de code reçu ne contiendra pas d'erreurs. Par contre, si le syndrome est non nul, on doit calculer le polynôme de localisation des erreurs et le polynôme d'amplitude des erreurs, à noter qu'il y a plusieurs méthodes de calcul de ces deux polynômes, dans le cadre de ce projet on utilisera une méthode, c'est le décodage selon l'algorithme d'Euclide, une fois les polynômes calculés en utilisant l'algorithme de Forney, on calculera les valeurs à soustraire pour obtenir le mot de code sans erreurs [18].

2.6.2.2 Calcul du Syndrome

Le calcul du syndrome est défini comme le reste de la division entre le polynôme reçu *r(x)* et le polynôme générateur *g(x),* tandis que le reste nous indique la présence d'erreurs, tous comme l'opération de la division, qui est toujours une opération complexe par rapport à des sommes et des additions, nous sommes amené à rechercher une autre méthode pour le calcul du syndrome [19], à cet effet, le calcul du syndrome peut aussi être effectué par un processus itératif, alors avant de pouvoir calculer le polynôme du syndrome, on doit attendre que l'on ait reçu tous les éléments du polynôme *r* (*x).*

$$S_i=r(\alpha^i)=c(\alpha^i) + e(\alpha^i) = e\ (\alpha^i)$$

On peut définir les différentes équations, à partir de cette relation, comme indiqué ci-dessous:

$$S_1 = e_{j1}\alpha^{j1} + e_{j2}\alpha^{j2} + \ldots\ldots\ldots + e_{jv}\alpha^{jv}$$

$$S_1 = e_{j1}\alpha^{2j1} + e_{j2}\alpha^{2j2} + \ldots\ldots\ldots + e_{jv}\alpha^{2jv}$$

$$S_{2t} = e_{j1}\alpha^{2tj1} + e_{j2}\alpha^{2tj2} + \ldots\ldots\ldots + e_{jv}\alpha^{2tjv}$$

Ce syndrome sous forme polynomiale est de la forme suivante:

$$S(x) = \ldots + S_{2t+1}x^{2t} + S_{2t}x^{2t-1} + \ldots\ldots + S_2x + S_1 \qquad \textbf{(2.31)}$$

Les premiers 2*t* symboles du syndrome sont les seuls connus, si le code *r* (*x*) n'est pas affecté par des erreurs, à cette effet, tous les coefficients du syndrome seront nuls (*r*(*x*) = *c*(*x*)).

Le schéma suivant, indique le calcul du syndrome d'une manière itérative :

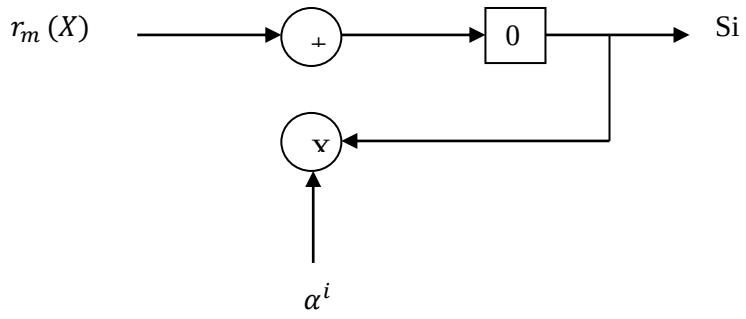


Figure 2.7 : Schéma pour calculer le syndrome.

2.6.2.3 Algorithme d'Euclide

2.6.2.3.1 Théorème d'Euclide« Généralités »

C'est un algorithme [22] récursif, permet de trouver le plus grand diviseur commun de deux polynômes $r_0(x)$ et $r_1(x)$ dans le champ de Galois GF(q), il existe deux polynômes $\alpha(x)$ et $b(x)$ en GF(q) tel que :

$$MCD\,(r_0(x), r_1(x)) = \alpha(x)\, r_0(x) + b(x)\, r_1(x) \qquad \textbf{(2.32)}$$

NB : on pourra calculer $\alpha(x)$ et $b(x)$ selon l'algorithme d'Euclide, tous en donnant deux polynômes non nuls $\alpha(x)$ et $b(x)$ en GF(q).

L'algorithme d'Euclide fonctionne de la façon suivante :

$$\deg(r_1(x)) \leq \deg(r_0(x))$$

$$\alpha_0(x) = 1, b_0(x) = 0 \qquad \textbf{(2.33)}$$

$$\alpha_1(x) = 0, b_0(x) = 1$$

Avec :

$\deg(r_1(x))$: Degré du polynome de $r_1(x)$;

$\deg(r_0(x))$: Degré du polynome de $r_0(x)$;

pour i ≥ 2, on va calculer le polynôme restant dont la division est effectuée par $r_{i-2}(x)$ et $r_{i-1}(x)$.

$$r_{i-2}(x)= q_i(x)\, r_{i-1}(x) + r_i \quad \textbf{(2.34)}$$

avec :

$$0 \leq \deg(r_1(x)) < \deg(r_{i-1}(x))$$

$$\alpha_0(x) = \alpha_{i-2}(x) - q_i(x)\alpha_{i-1}(x) \quad \textbf{(2.35)}$$

$$b_1(x) = b_{o-2}(x) - q_i(x)\, b_{i-1}(x)$$

Lorsque : deg (r_i) = 0, les calculs se terminent, le dernier polynôme non nul, indique le plus grand diviseur commun [20].

2.6.2.3.2 Correction des erreurs « Euclide »

Le polynôme de localisation d'erreurs est défini comme dans [19,22] par :

$$\sigma(x) = \prod_{k=1}^{v}(1 - \alpha^{ik} x)$$

$$= \sigma_v x^v + \sigma_{v-1} x^{v-1} + \ldots\ldots + \sigma_1 x + 1$$

Le polynôme d'amplitude des erreurs est calculé de la manière suivante :

$$\omega(x) = S(x)\sigma(x) \quad \textbf{(2.36)}$$

- $\sigma(x)$: c'est le polynôme de localisation des erreurs ;
- S(x) : c'est le polynôme syndrome ;
- ω (x) : c'est le polynôme d'amplitude, " inconnu à ce stade ".

On connaît que 2*t* symboles du polynôme du syndrome ($x^0 \ldots . x^{2t-1}$), donc, on devrait limiter le résultat à 2*t* tel que:

$$S(x)\, \sigma(x) = \omega(x) \bmod (x^{2t}) \quad \textbf{(2.37)}$$

Cette fonction est l'équation clé pour les codes de Reed-Solomon, si le nombre d'erreurs *v* dans le mot de code transmis C(x) est plus petit ou égale à *t*, donc, l'équation clé a une seule paire de solution $\sigma(x)$ et ω (x), les deux degrés des polynômes doivent respecter la contrainte :

$$\deg\big(\omega(x)\big) < \deg\big(\sigma(x)\big) < t \quad \textbf{(2.38)}$$

L'équation clé résolue selon l'algorithme d'Euclide en appliquant $r_0(x) = x^{2t}$ et $r_1(x)$ et $r_1(x) = S(x)$.

Le calcul du théorème d'Euclide, il donne une solution de polynôme de localisation des erreurs et le polynôme [18].

Le reste dernier de la division, il donne le polynôme d'amplitude, dont le polynôme de localisation des erreurs est donné selon la relation :

$$\boldsymbol{\sigma}_i(x)= \boldsymbol{\sigma}_{i-2}(x) + \boldsymbol{\sigma}_{i-1}(x)Q_i(x) \quad \textbf{(2.39)}$$

D'où :$_i(x) = B_i(x)$

NB: Dans cette théorie, on est obligé d'avoir deux blocs dans l'implémentation hardware, primo : un bloc qui effectue la division et qui donnera le polynôme d'amplitude des erreurs, et secondo : un bloc de multiplication qui donnera le polynôme de localisation des erreurs [20].

2.6.2.4 Le Chien Search « CS »

On doit évaluer les racines de polynôme de localisation, une fois le polynôme de localisation des erreurs calculé, l'évaluation des racines est effectuée avec l'algorithme appelé « **Chien Search**» qui est du type « brute force », c'est-à-dire, qu'il évalue toutes les solutions possibles.

À la sortie de ce bloc, on obtiendra une séquence de symboles, et lorsque ces derniers seront nuls; ceci nous indique qu'une racine a été détectée [20].

2.6.2.5 Algorithme de Forney

L'algorithme de Forney, permet de construire le polynôme d'erreurs $e(x)$ et l'additionner avec le polynôme reçu $r(x)$ pour reconstruire le polynôme $c(x)$, pour le calcul du polynôme $e(x)$, les polynômes $\boldsymbol{\sigma}(\alpha')$, $\boldsymbol{\sigma}'(\alpha')$, $\omega(\alpha')$sont nécessaires, à savoir, le polynôme de localisation des erreurs et sa dérivée qui sont déjà évalués pour les différentes valeurs de α, il reste qu'à évaluer et une fois les différentes valeurs calculées, on applique l'algorithme de **Forney**[18], qui est définit comme suit:

$$e_i = \frac{\omega(\alpha^i)}{\sigma'(\alpha')} \quad \textbf{(2.40)}$$

$\omega(\alpha^i)$: Polynôme d'amplitude évalué pour les valeurs de $GF(2^4)$.

$\sigma'(\alpha')$: Dérivée du polynôme de localisation des erreurs pour les valeurs de $GF(2^4)$.

2.6.3 Correction des effacements

Les codes de Reed-Solomon RS sont utilisés pour la correction des erreurs, et aussi pour corriger les effacements, qui suit le même principe que lorsqu'on efface une lettre dans un mot à l'aide d'un effaceur, cette dernière est dans le mot n'est pas connue, mais la position de celle-ci l'est, onc, les codes de Reed-Solomon permettent de corriger deux fois plus d'effacements que d'erreurs, dont la séquence de décodage est la même que celle utilisée pour la correction des erreurs, mais, la seule différence est qu'avant de calculer les syndromes, on doit substituer dans le polynôme reçu $r(x)$ les effacements avec des '0' avant de procéder au calcul du syndrome lui-même, à savoir que la première opération à effectuer pour le décodage des erreurs et des effacements est l'évaluation du polynôme de localisation des effacements [18].

On doit signaler que la capacité de la correction sera plus grande que dans l'absence des effacements lorsqu'on devrait ajouter f effacements aux *v* erreurs.

2.6.4 Probabilité et le taux d'erreur

Pour un code de canal, il existe deux mesures importantes [23]:

2.6.4.1 Taux d'erreur Binaire « BER »:

Le taux d'erreur binaire constitue le paramètre primaire décrivant la qualité de la transmission numérique, d'où, il est définit comme le rapport entre les bits erronés et le nombre total de bits reçus, ce taux détermine le nombre d'erreurs apparues avant la modulation et après la démodulation, il augmente suite à des perturbations,

à savoir : équipement ou réseau défectueux, pointage incorrect d'une antenne et longueur de canal [24].

2.6.4.2 Probabilité d'erreur :

Les codes avec une probabilité d'erreur inférieure sont plus souhaitables. Pour les codes de correction d'erreur, la probabilité d'erreur de bloc est importante. Pour les codes de détection d'erreur, la probabilité d'erreur non détectée est importante [23].

2.6.5 Probabilité d'erreur de « bit/bloc »

2.6.5.1 Pe en bit :

C'est la probabilité d'erreur de bit entre le message original et le décodé [23], qui exprime le taux des bits erronés en référence du message original.

2.6.5.2 Pe en bloc :

C'est la probabilité d'une erreur de décodage du mot reçu en fonction de bloc, à cet effet, il est probable que le décodeur choisit le mot de code non correct lors de l'application de l'algorithme de décodage [23].

Exemple 2.10 :

On considère que le message 000 est transmis sous forme de mot de code 000000, des erreurs de deux bits se produisent dans les deux derniers bits et le mot reçu est 000011, le décodeur de canal sélectionnera ensuite le mot de code le plus proche qui est 100011, et une erreur de bloc s'est produite, si cela se produit toujours, on est à faire à une et cependant, le mot de code 100011 est décodé en tant que message 100 qui n'a que le premier bit le plus faible et les restes deux bits sont corrects selon le message original, alors :

$$\mathrm{Pb(kn)} = \frac{1}{3} = 0.33 \qquad \textbf{(2.41)}$$

2.7 Conclusion

De ce qui précède, on a vu dans la première partie de ce chapitre, le principe de fonctionnement du codage Reed-Solomon, où on a décrit toutes les étapes essentielles pour les opérations de codage et de décodage avec simulation sur Matlab, et on a conclu que les codes RS offrent une grande capacité dans la détection et la correction des erreurs ainsi que les effacements, dont la chose qui priorise d'avantage ce code pour l'employer actuellement dans presque toutes les applications multimédia, transmissions numériques et spatiales en précisant, particulièrement, les communications à base de protocoles à accès aléatoire où les taux d'erreurs sont plus importants.

CHAPITRE 3 :

Evaluation de la performance du code de Reed-Solomon « RS »

3.1 Introduction

Les paramètres du codage et le décodage et les conditions du support de transmission rapportent sur les performances du codage RS employé, dans les deux chapitres précédents, on a pu exposer le principe de fonctionnement du codage (Reed-Solomon) et ses caractéristiques vis-à-vis les différents types de canal.

Dans le même contexte, on va maintenant examiner le comportement du codage (RS) par rapport à la variation de ses à différents paramètres, c'est ce qu'on va le voir dans le dernier chapitre 3.

3.2 Probabilité d'erreur (Code de correction d'erreur t-bit) :

Probabilité d'erreur en bloc pour un code de correction d'erreur t-bit :

La probabilité d'erreurs en bloc est exprimée par l'équation suivant [23]:

$$P_e(kn) = 1 - \sum_{i=0}^{t} \binom{n}{i} q^i (1-q)^{n-i} \quad \textbf{(3.1)}$$

Kn présente un codeur de « n » bits, tel que la probabilité d'erreurs de *i* bits erronés entre n bits du mot de code est égale à :

$$\binom{n}{i} q^i (1-q)^{n-i} \quad \textbf{(3.2)}$$

Un code correcteur d'erreurs t-bit est capable de gérer des erreurs de t bit, à cet effet, la probabilité erreur de décodage se produit est [23] :

$$Pc(kn) = \sum_{i=0}^{t} \binom{n}{i} q^i (1-q)^{n-i} \quad \textbf{(3.3)}$$

La probabilité d'erreur est comme suit:

$$P_e(kn) = 1 - Pc(kn) \quad \textbf{(3.4)}$$

On suppose qu'un canal CBS avec q=0.001, dans le cas d'un canal sans code correcteur ; on a :

$$P_b(K_1) + 0{,}001 = 1 \text{ X } 10^{-3} \quad \textbf{(3.5)}$$

Le Rendement du code = m/n toujours inférieur de 1, R≈1 le code est efficace. **(3.6)**

On utilise un mot de code de n=3, afin de valider les équations précédentes, par un code correcteur de 1 bit, et on aura ce qui suit :

$$P_e(k_3) = 1 - \sum_{i=0}^{t} \binom{n}{i} q^i (1-q)^{n-i} \quad \textbf{(3.7)}$$

$$= 1 - \binom{3}{0} (0.999)^3 - \binom{3}{1} (0.999)^2 (0.001)^1$$

$$= \mathbf{3 \text{ X } 10^{-6}}$$

À partir de k=1, on aura :

$P_b(K_3) = P_e(K_3) =$ **3 x 10 $^{-6}$** **(3.8)**

On constate que la probabilité est diminuée de1 X 10^{-3} à 3 X 10^{-6}, même, pour le taux de codage qui a diminué de 1 à $\frac{1}{3}$ [23].

Nous allons maintenant évaluer la variation de la probabilité d'erreurs en fonction de nombre de bits erronés d'une façon plus généralisée en faisant des simulations par Matlab ; afin de voir encore le comportement du codage RS.

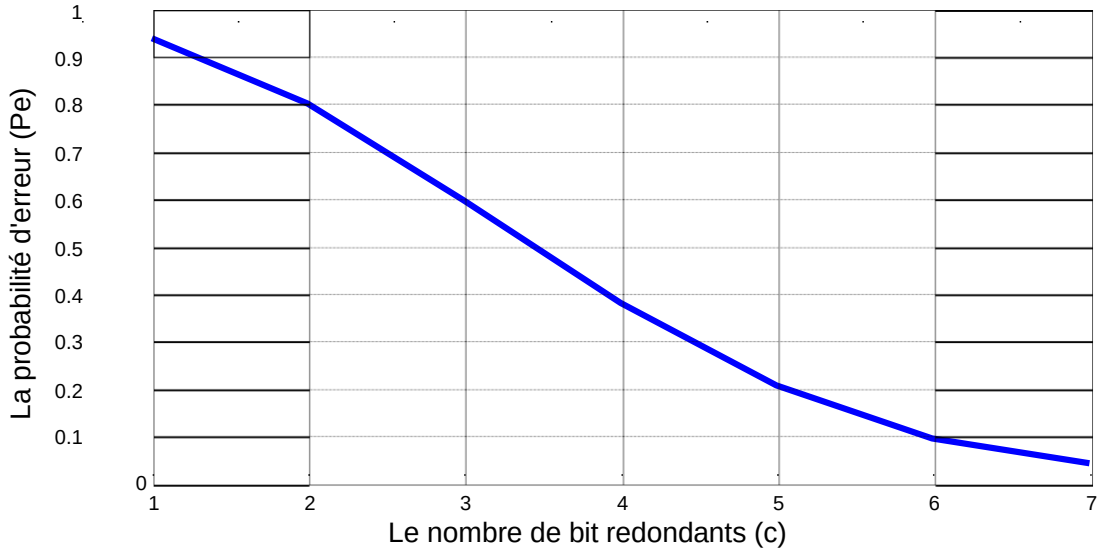


Figure 3.1: *La variation de la probabilité d'erreur en fonction du nombre de bit de redondance* ***c***.

Dans la figure ci-dessus « **figure 3.1** », qui représente la variation de la probabilité d'erreur en fonction des bits de redondances ***c*,** tel que la probabilité d'erreur varie inversement proportionnel par rapport à la variation du nombre de bits redondance,

on constate que la probabilité d'erreurs (P_e) atteint son valeur maximale **(**presque≈**0.98**), lorsque le nombre de bits redondants égale à 1, et prend en diminution pour atteindre sa valeur minimale (**jusqu'à 0.05**) lorsque le nombre de bits redondant égale à 7, à cet effet, on conclue que la capacité de correction dans le codage de Reed-Solomon, devienne plus importante avec un nombre de bits de contrôle plus important aussi, à l'issu, plus le nombre des bits redondants est supérieur plus la capacité est meilleure.

Cependant, l'augmentation du nombre de bits redondants affaiblie en contrepartie la vitesse de transmission lorsque la quantité des bits transmis sera importante. En ajoutant à cela la complexité des opérations de codage et décodage qui peut nuire aux performances du codeur.

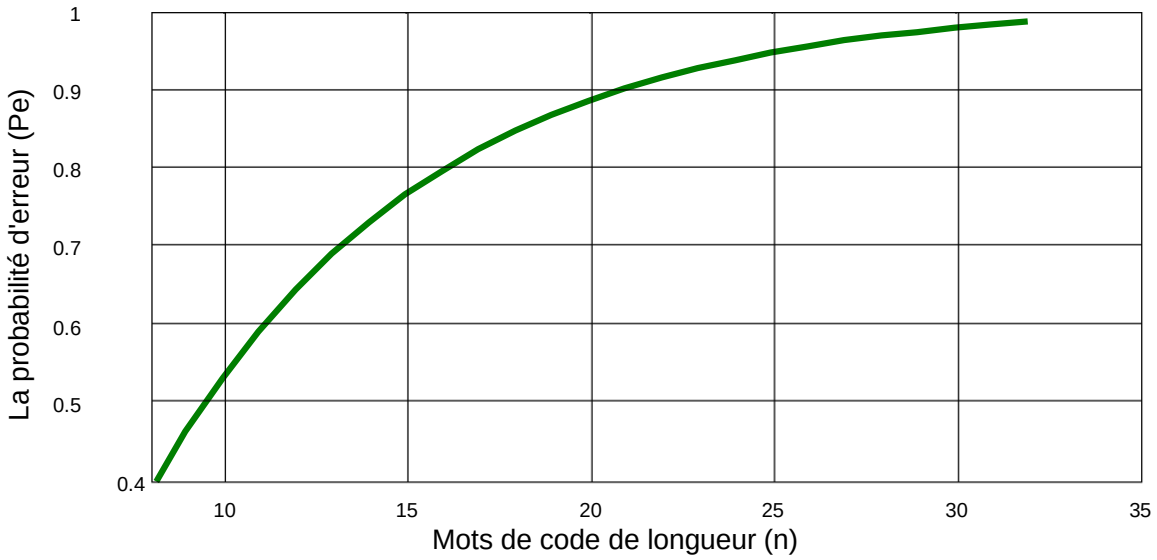


Figure 3.2 : *La variation de la probabilité d'erreur en fonction de la longueur* ***n*** *du mot de code.*

La figure 3.2 représente la variation de la probabilité d'erreur en fonction de mots de code **(n).** On remarque qu'il y a une relation proportionnelle entre et ***n.*** Tel que pour des longueurs petites des mots de code la probabilité d'erreurs est aussi faible. Tandis que pour des longueurs plus grandes la probabilité devient maximale et prend sa stabilité malgré l'augmentation de ***n***. Nous pouvons conclure qu'avec des mots de code de longueurs élevées le codage porte alors de faibles performances, ce qui aggrave par conséquence la transmission des données lorsque la capacité de correction devient plus faible.

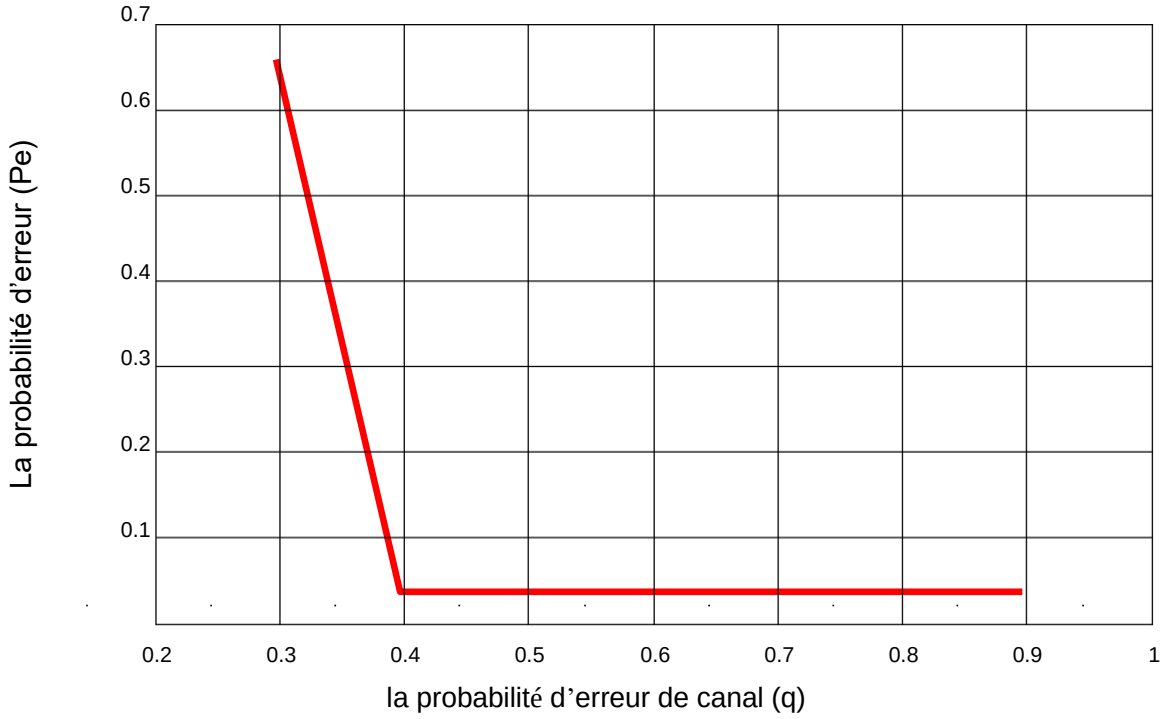


Figure 3.3 : *La variation de la probabilité d'erreur en fonction de la probabilité d'erreur de canal* ***q***

La figure 3.3 représente la variation de la probabilité d'erreur en fonction de la probabilité d'erreurs du canal, qui décrit une variation inversement proportionnelle, à savoir les faibles valeurs de ***q***, on a des P_e maximales., à partir de q=0.4 on enregistre une stabilité de la variation confirme la validité de l'équation (3.1) qui explique la relation non proportionnelle entre le **q** et le P_e, à cet effet, le code Reed Solomon présente une forte immunité contre les milieux bruités, pour des canaux présentant un taux d'erreurs élevé le code Reed-Solomon offre une probabilité d'erreurs plus faible et une probabilité de succès plus élevée, à savoir que l'article dans [25] présente la probabilité d'erreurs en bloc comme suit :

$$P_e \approx \frac{1}{2^m-1} - \sum_{i=t+1}^{2^m-1} i\binom{2^m-1}{i} q^i (1-q)^{2m-1-i} \qquad \textbf{(3.9)}$$

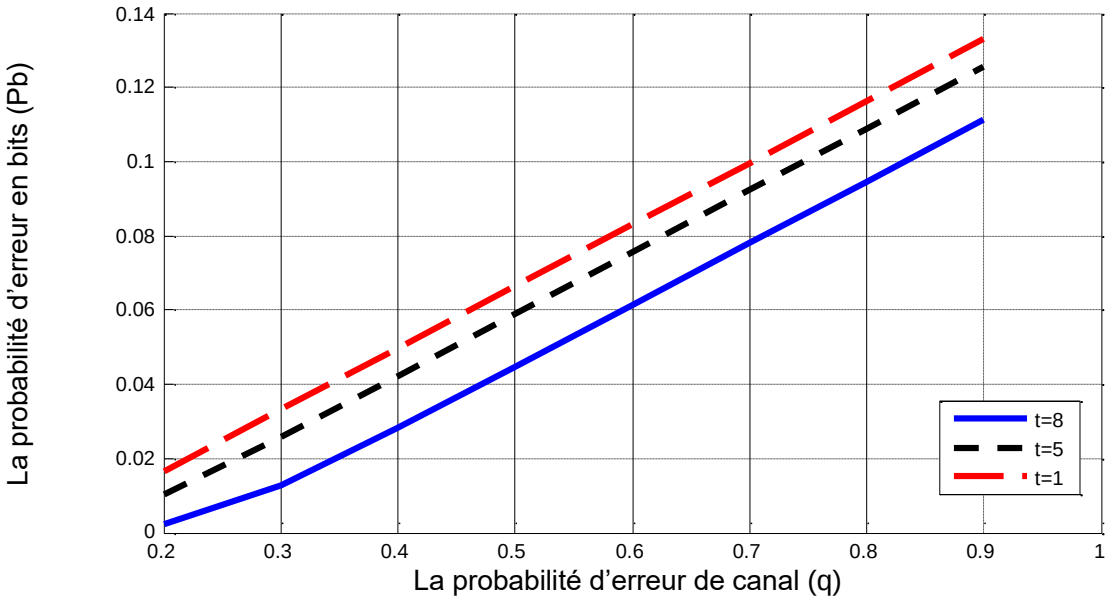
n= 2^m-1, alors :

$$P_e \approx \frac{1}{n} - \sum_{i=t+1}^{n} i\binom{2^m-1}{i} q^i (1-q)^{n-i} \qquad \textbf{(3.10)}$$

Où, Le **t** est la capacité de correction d'erreur, et les symboles sont constitué de **m** bits chacun.

La probabilité d'erreur de bit peut être délimitée par la probabilité d'erreur de symbole pour un type de modulation spécifique. Pour la modulation MFSK avec ***M*** = , la relation entre et est la suivante [25] :

$$\frac{P_\beta}{P_e} = \frac{2^m-1}{2^m-1} \qquad (\mathbf{3.11})$$



La figure 3.4 : *La variation de la probabilité d'erreur en bit en fonction de la probabilité d'erreur de canal* ***q***.

La figure 3.4 représente la variation de la probabilité d'erreur en bits en fonction de la probabilité d'erreur du canal ***q***. Nous observons que la variation de augmente parallèlement avec l'augmentation de ***q***, tel que pour une valeur minimum de **q** la probabilité d'erreur est faible et augmente relativement à l'augmentation de ***q***. La simulation montre aussi que la probabilité d'erreurs augmente lorsque le nombre de bits erronés par mot est augmenté. Alors pour avoir de bonnes performances de la transmission ; il est préférable de choisir un canal de transmission plus adéquat et qui offre une faible ***q.***

3.3 Conclusion

Les performances du codage (décodage) RS sont analysées dans ce chapitre. Nous avons entamé des études par des simulations multiples pour pouvoir décrire les caractéristiques d'un codeur/décodeur RS, afin d'évaluer à la fin son comportement, sa capacité et son efficacité dans des milieux de propagation plus au moins bruités. En effet, l'ensemble des simulations faites au sein de ce chapitre nous a permet de savoir comment le codage RS devient plus efficace par un choix plus convenable de nombre de bits redondants et donc la longueur du mot de code suivant la qualité du canal de transmission. Et cela exprime également la capacité de correction chez le décodeur RS. Cette capacité qui est mesurée par la probabilité d'erreurs binaire notée (P_β), et représentée en pratique par le taux d'erreurs binaire.

En fait, nous découvrons une efficacité importante du codage (décodage) RS que présente en particulier lors des canaux de transmission qui ont un taux d'erreurs élevé. Cependant la formulation du mot code est assez délicate et fait créer une sensibilité considérable du codeur (décodeur) RS. Chose qui exige une certaine mesure avant de prendre décision sur la sélection du mot de code en prenant en considération le type de milieu dans lequel la transmission est déroulée pour éviter l'affaiblissement des performances de la transmission lorsque la capacité et le débit serons affaiblies à cause des retransmissions multiples pour réussir la transmission.

Conclusion générale

Le codage du canal de communication introduit systématiquement une redondance et donc une augmentation du flux des données en ajoutant des bits au message initial de telle façon que la détection et la correction des erreurs soit faites aisément. L'association du codage et du décodage du canal de communication a pour tâche de minimiser les effets du bruit dans ce canal.

Notre objectif dans cet humble travail de mémoire de master est d'étudier et d'analyser un des techniques de codage de canal qui le code de Reed-Solomon qui est très utilisé en pratique pour détecter et très utilisé en pratique pour détecter et corriger plusieurs erreurs à la fois.

Dans le chapitre 2, nous avons expliqué comment fonctionne le code de Reed-Solomon avec des exemples et simulation Matlab.

Au chapitre 3, nous avons analysé les performances de ce codeur R S.

Ce travail nous a permis d'avoir des connaissances en théorie de l'information et codage pour compléter notre formation.

En perspective, nous souhaitons que ce code de R.S soit implémenté sur DSP ou FPGA

Les références

[1] M. Côte,"Reconnaissance de codes correcteurs d'erreurs," Thèse de Doctorat, Ecole Polytechnique en Informatique,2010

[2] J.G. Proakis, M. Salehi, Digital Communications, 5th ed. McGraw-Hill,2008

[3] G. BERHAULT, "Exploration architecturale pour le décodage de codes polaires," Thèse de Doctorat, université de Bordeaux, France,2015

[4] M. CUNCHE, "Codes AL-FEC hautes performances pour les canaux à effacements : variations autour des codes LDPC," Thèse de Doctorat, Ecole Doctorale Mathématiques, Sciences et Technologies de L'information(Grenoble),2010

[5] G. LANDAIS, "Mise en œuvre de crypto systèmes basés sur les codes correcteurs et de leurs cryptanalyses," Thèse de Doctorale, Université Pierre et Marie Curie -Paris VI,2014

[6] M. Demazure, Cours d'algèbre, Cassini1997

[7] X. HENOCQ," Contrôle d'erreur pour transmission de flux vidéo temps réels sur réseaux de paquets hétérogènes et variant dans le temps," Thèse de Doctorat, Ecole Mathématique, Télécommunication, Informatique, Système Electroniques (Matisse) ,2002

[8] C. CHRISTOPHE, "Reconnaissance de codes structure des codes quasi-cycliques," Thèse de Doctorat, École Doctorale Science–Technologie-Sante Faculté des Sciences et Techniques XLIM–DMI Equipe PI2C (Limoges), 2009

[9] A.IRINA, "Code correcteurs d'erreurs LDPC structurés, "Thèse de Doctorat, Université Laval, France, 2010

[10] J. HOUSSEIN, "Conception architecturale haut débit et sûre de fonctionnement pour les codes correcteurs d'erreurs," Thèse Doctoral, École Doctorale IAEM - Lorraine,2009

[11] R. Tragi, P.C. Srivastava, M. Kumar, R.K. Singh, "Performance Comparison of RS code and Turbo Codes for Optical Communication," Serials Publications, ISSN: 0973-7383.vol.3, no.2, pp.251-256,2011

[12] I. DIOP, A. DIOP, I. DIOUM, S.M. FARSSI, K. TALL, S. OUYA, "Etude de la Performance des Codes de Reed Solomon et Turbo Code dans la Compression JPEG 2000 en Environnement Bruite, " J.sci.vol.10, N°. 2, pp.5-12,2010

[13] R. w. Hamming, "Error detecting and Error correcting codes, "Bell system Tech.pp.147-160,1950

[14] Y. Jiang," A Practical Guide to Error-Control Coding Using MATLAB", U.S. Library of congress, Artech house Boston/London artechhouse.com, 293p.ISBN 13: 978-1-60807-088-6, 2010

[15] O.M. Mohamed Bouye," Application des codes correcteurs d'erreurs en Sténographie," Thèse de Doctorat, Université Mohammed V, Rabat,2009

[16] S. Dietler, " Implémentation de codes de Reed-Solomon sur FPGA pour communications spatiales," Thèse de Doctorat, Université de Larraine,2006

[17] W. Stalling, "Cryptography and Network Security", Third edition, Prentice Hall,2003

[18] J. A. Buchmann, "Introduction to Cryptography, "Second edition, October2003

[19] C. S. Mondlin, "Error control coding in DSL system, CRC Press LCC, "2004

[20] M. L. Boucenna, "Protocoles de Communications Satellitaires avec Sécurisation de la transmission et Récupération des Données, "Thèse de Magister, Université de Constantine Faculté des Sciences de l'Ingénieur, Algérie,2008

[21] V. Thierry, "Recherche des performances dans la mise en œuvre des codes linéaires cycliques en ASIC à haut débit," Thèse de Doctorat, Université de Metz et Supelec,1999

[22] C. Lee, "Error-Control Block Codes," Artech House Publishers,2000

[23] R. Togneri, Ch. J.S. de Silva, "Fundamentals of information theory and coding design", Thèse publiée dans Taylor & Francis e-Library, 398p.ISBN 1-58488-310-3,2006

[24] F. NABILA, H. SOUHAILA, "Etude et simulation d'une transmission de type OFDM pour les communications sans fil, " Thèse de Doctorat, Université Larbi Tebessi, Tebessa, Algérie,2016

[25] B. SKLAR "Digital communications fundamentals and Applications ", Second Edition, Tarzana, California and University of California, Los Angeles, 2001.1032p.ISBN0-13-084788-7

[26] Syed Shahzad Shah, SaqibYaqub, and Faisal Suleman, Self-correcting codes conquer noise Part 2: Reed-Solomon codecs, Chameleon Logics, (Part 1: Viterbi Codecs), 2001.

[27] S. B. Wicker, Error Control Systems for Digital Communication and Storage, N.J.: Prentice-Hall, 1994.

[28] Y. Sugiyama, Y. Kasahara, S. Hirasawa, and T. Namekawa, "A Method for Solving Key Equation for Goppa Codes," Information and Control, Volume 27, pp. 87-99, 1975.

[29] Ling-Pei Kung, "Introduction To Error Correcting Codes", NSDL Scout Report for Math, Engineering, and Technology, 2003.

[30] RaghavBhaskar, Pradeep K. Dubey, Vijay Kumar, AtriRudra, "Efficient Galois Field Arithmetic On SIMD Architectures", ACM Symp. On ParallelAlgorithms and Architectures, pp 256-257, 2003.

[31] T.K. Truong, L.J. Deutsch, I.S. Reed, I.S. Hsu, K. Wang, and CS. Yeh, 'sThe VLSI Design of a Reed-SoIomon Encoder Using Berlekamp's Bit-Senal Multiplier Algorithm," Third Caltech Conf. on VLSI, pp. 303-329, 1983.

[32] I.S. Hsu, I.S. Reed, T.K. Truong, K. Wang, CS. Yeh, and L.J. Deutsch, "The VLSI Implementation of a Reed-Solomon Encoder Using Berlekamp's Bit-Serial Multiplier Algorithm," IEEE Trans. Comput., vol. (3-33, no. 10, pp. 906-911, Oct. 1984.

[33] James S.Plank, A Tutorial on Reed-Solomon Coding for Fault- Tolerance in RAID- like Systems, Technical Report CS-96-332.

[34] M .A. Hasan and V. K. Bhargava, "Division and Bit-Serial Multiplication over GF(2m)," IEE Proc., part E, vol. 139, no. 3, pp. 230-236, May 1992.

[35] G. Orlando and C. Paar, “A Super-Serial Galois Fields Multiplier For Fpgas And Its Application To Public-Key Algorithms”, Proc. of the 7th Annual IEEE Symposium on Field Programmable Computing Machines, FCCM’99, Napa Valley, California, pp. 232-239, April. 1999.

[36] J. L. Massey, "Shift Register Synthesis and BCH Decoding," IEEE Transactions on Information Theory, Volume IT-15, Number 1, pp. 122- 127, January 1969.

[37] Favalli M., Metra C., “Optimization Of Error Detecting Codes For The Detection Of Crosstalk Originated Errors”, Design Automation and Test in Europe, pp 290-296, March 2001.

[38] C.E. Shannon, “A Mathematical Theory Of Communication”, Bell System Technology Journal, volume 27, pp. 379-423, 623-656, 1948.S