AN INVESTIGATION INTO JAMMING GSM SYSTEMS THROUGH EXPLOITING WEAKNESSES IN THE CONTROL CHANNEL FORWARD ERROR CORRECTION SCHEME.

A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Masters of Science in Engineering (Electrical).

Gareth Timm - Student No. 481195

Research Supervisor: Prof. Jaco Versfeld

Johannesburg, 2017

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily the be attributed to the NRF.

Declaration

I declare that this dissertation is my own, unaided work, except where otherwise acknowledged. It is being submitted for the degree of Master of Science in Electrical Engineering to the University of the Witwatersrand, Johannesburg, South Africa. It has not been submitted before for any degree or examination at any other university.

Candidate Signature:	....
Name:	....
Date:	(Day)...............(Month)...............(Year).........................

Abstract

The ability to communicate effectively is of key importance in military scenarios. The ability to interfere with these communications is a useful tool in gaining competitive advantages by disrupting enemy communications and protecting allied troops against threats such as remotely detonated explosives. By reducing the number of corrupt bits required by using customised error patterns, the transmission time required by a jammer can be reduced without sacrificing effectiveness. To this end a MATLAB simulation of the GSM control channel forward error correction scheme is tested against four jamming methodologies and three bit corruption techniques. These methodologies are aimed at minimising the number of transmitted jamming bits required from a jammer to prevent communications on the channel. By using custom error patterns it is possible to target individual components of the forward error correction scheme and bypass others. A random error approach is implemented to test the system against random errors on the channel, a burst error approach is implemented to test the convolutional code against burst errors, and two proposed custom error patterns are implemented aimed at exploiting the Fire code's error detection method. The burst error pattern approach required the least number of transmitted jamming bits. The system also shows improvements over current control channel jamming techniques in literature.

Acknowledgements

This dissertation is dedicated to my mother who has shown me never ending love and support during the completion of my studies.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AGC	Automatic Gain Control
AGCH	Access Granted Channel
ARFCN	Absolute Radio Frequency Channel Number
BCH	Broadcast Channel
BSIC	Base Station Identity Code
CCCH	Common Control Channels
CRC	Cyclic Redundancy Check
DCCH	Dedicated Control Channel
FACCH	Fast Associated Control Channel
FCCH	Frequency Correction Channel
FEC	Forward Error Correction
GSM	Global System for Mobile Communication
IED	Improvised Explosive Device
MER	Message Error Rate
MS	Mobile Subscriber
NCH	Notification Channel
OSI	Open Systems Interconnection
PCH	Paging Channel
RACH	Random Access Channel
RFN	Reduced TDMA Frame Number
SACCH	Slow Associated Control Channel
SACCH	Standalone Associated Control Channel
SCH	Synchronisation Channel
SDCCH	Standalone Dedicated Control Channel
SMS	Short Message Service
SNR	Signal to Noise Ratio
TCH	Traffic Channel
TDMA	Time Division Multiple Access

List of Symbols

Symbol	Units	Description
$e_t(x)$	-	Error message to be introduced.
$e_i(x)$	-	Error message before interleaving.
$e_c(x)$	-	Error message before convolutional encoding.
$c_t(x)$	-	Error free message being transmitted.
$c_r(x)$	-	Received corrupted message.
$m_r(x)$	-	Received message after decoding.

Chapter 1: Introduction

1.1 Background

Cellular phones and the Global System for Mobile Communication (GSM) network is used globally as one of the main means of communication. The GSM network remains as the dominant technology in 2016 with the number of connections surpassing 7.7 billion [1]. With the abundant use of this technology it is not surprising that it finds uses in a wide variety of applications such as remote monitoring devices, tracking devices, and most commonly in mobile phones. Signal jammers are used to prevent these systems from being able to function correctly.

Signal jamming is where a custom (or random) signal is broadcast with the intent of interfering with another wireless communication channel, effectively making communication on that channel impossible. This is usually done by transmitting a more powerful random signal with the same carrier frequency to 'drown' out the original signal and make it unreadable. A major concern for all personnel in places of war is the Improvised Explosive Device (IED), which is a home-made explosive often triggered remotely using a radio detonator such as a mobile phone. The radio controlled IED can be monitored from far away and when the target is in range of the explosive, an SMS or call is made to the mobile detonator which causes the device to detonate [2]. Signal jammers reduce this risk by eliminating the possibility of communication on certain frequencies such that these calls or SMS's cannot be delivered to the mobile detonator.

In the most basic jamming approach this jamming signal is constantly transmitted on the channel at a high power, this is termed an 'always on' jammer [3]. The downfall to this approach is the requirements for a constant jamming signal to be transmitted as it creates a constant interference presence on the channel and requires a constant power source. This constant presence makes the jammer more easily detectable, and provides a large sample collection for direction finding methods to be used to locate the device. For example, if the army is using signal jammers to block remote explosives, it would be detrimental for the enemy to be able to locate and disable the jammer. If the duration for which the signal is transmitting can be reduced, the detectability and traceability of the device is also reduced resulting in a safer environment for those using the device [4].

It is seen that the usage of signal jammers can be a great benefit in protecting users, a downfall to these signal jammers is in situations where they also block legitimate or crucial communications such as calls to emergency services. This has led to the usage of these devices by the public being made illegal in most countries [5]. They are still widely used in many military situations to protect troops and gain communication advantages over adversaries. The large power requirements from these jammers to achieve successful jamming is not practical when they are used in situations without access to a constant power source. Therefore if the devices are used in locations which are not easily accessible such as spread around an enemy battlefield, then they are required to be independently powered [6]. This leads to the need for energy efficient signal jammers to be developed to ensure the duration for which they can be used is sufficient when there is no constant power supply available.

1.2 Problem Identification

Due to signal jammers being used in places of war such as on a battlefield, it is not always possible for these jammers to have access to constant power sources, which results in them having to use limited power sources such as battery packs. Due to the high energy requirements of signal transmission [7] which increases with the range of jamming required, the lifetime of these systems is limited by their available power supply. Thus the need for energy efficient signal jamming systems exists to extend the duration for which these systems can operate, the first approach to achieving this is reducing the required transmission time from the jammer.

A reduction in the overall required transmission time of the jamming system can be achieved through a reduction in the number of bits requiring transmission to destroy the original message. By reducing the number of bits requiring transmission, and hence the required jamming time, this allows for a reduction in the energy requirements of the jammer and for the system to be less detectable due to reduced presence on the channel.

This leads us to the following research question:

Research Question: "How can we exploit the control channel forward error correction scheme of the GSM system in order to minimize the number of jamming bits required to prevent communications on the channel?"

To be able to answer this research question, it is important that we define some key terminology to be used throughout the remainder of this research.

1.2.1 Important Definitions and Assumptions

Communication failure: The inability for any device on the network to initiate or receive any further calls or SMS's, this does not extend to disrupting any existing active connections.

Message error rate: The ratio of the number of incorrect messages received (decoding errors) over the total number of messages received.

Communication: This is limited to calls and SMS's and does not extend to data transmission on the GSM network.

Jamming bits: The number of transmitted bits required by the jamming system.

Assumptions: This research focusses on the data link layer, and as such to be able to make a fair comparison on the bit level, we assume a very high Signal to Noise Ratio (SNR) on the channel for the original signal. By assuming high SNR, the effects of channel interference such as AWGN and multipath fading can be ignored, and the focus of the investigation can be placed on the individual jamming patterns. We also assume that the required time for a jammer to corrupt a bit is the same duration as one GSM bit.

1.3 Scope and Research Objectives

This research aims to undertake an in-depth study of the GSM network protocol and the forward error correction (FEC) schemes in use, and through its understanding verify the feasibility of improved control channel jamming methodologies aimed at exploiting weaknesses in these error correction schemes. From this a set of improved jamming strategies is developed in attempt to minimise the overall transmission time required to effectively prevent successful communication on the channel. In this research "communication" is limited to only calls and SMS's and does not apply to data connections. This is to be done by using an "intelligent" control channel jamming methodology (one which focusses on the data link layer of the OSI model) using the MATLAB simulation software to simulate the GSM Common Control Channel (CCCH) and the jamming system. This GSM system is then tested against a variety of jamming techniques (discussed in Section 3.2) focussed specifically on the GSM CCCH forward error correction scheme, instead of the whole signal as in other jamming methodologies (discussed in Section 2.3). The results obtained are to be contrasted amongst each other and compared with recent research done up until the time of writing this document.

To be able to answer the research question it is important that we define a specific parameter according to which all the jamming methodologies are compared. In this research the jamming methodologies are compared according to the total number of bits required to be jammed (corrupted) to cause communication failure on the channel. Communication failure is defined in this research as when no further calls or SMS's can be successfully executed, and does not attempt to interrupt existing connections. The analysis is done on a bit level, and as such it is straight forward to calculate the transmission time required from the jammer. The transmission time is calculated as the number of bits which are jammed multiplied by the transmission time per bit. This can be done as follows:

$$t_{trans} = n_{bits} * \frac{15/26}{156.25}\, ms \tag{1}$$

The transmission time per bit is calculated as the transmission time for one timeslot (15/26 = 0.577ms) divided by the number of bits per timeslot (156.25 bits) [8].

1.4 Dissertation Organization

This section provides insight into the layout and content present in this dissertation. The chapter layout is shown below, with a brief overview of each chapter given.

1.4.1 Literature Review (Chapter 2)

Chapter 2 contains an in-depth literature review section which is dedicated to exploring all the relevant details of the GSM system critical to the success of this research, as well as providing insight into the common jamming methodologies being used today. Similar research done prior to this work which is relevant to this research is also discussed outlining the methods, contributions, and results obtained. The chapter begins by discussing the necessary components of the GSM system such as the frame structure, channel structure, and connection setup information for the system, as well as the forward error correction and time diversity techniques being used. The chapter then outlines the four common jamming techniques used in practice today, and explores similar research papers attempting to improve on these jamming techniques for various different wireless communication protocols, outlining the achievements made in each.

1.4.2 Research Methodology (Chapter 3)

Chapter 3 contains an in-depth explanation of the various jamming methodologies tested during the completion of this research. The chapter begins by discussing the software requirements used in the testing procedures, as well as the three different bit corruption techniques used for testing the jamming methodologies. Each of these jamming methodologies are discussed in detail with reasons for choices given, the chapter concludes by suggesting two proposed jamming methodologies which attempt to improve on current jamming techniques.

1.4.3 Results from Individual Jamming Methodologies (Chapter 4)

Chapter 4 presents the results obtained during the testing procedures outlined in Chapter 3. These results are first analysed individually highlighting key points in each, then at the end of the chapter each of the jamming methodologies are compared, first with one another then with recent results from literature.

1.4.4 Conclusion (Chapter 5)

Chapter 5 is the final chapter. In this chapter the research is summarised, followed by a discussion on the achievements made. A short conclusion based on the results obtained is then presented and the chapter concludes by suggesting possible future directions in which this research can proceed.

Chapter 2: Literature Review

To complete this research there are two main systems which are implemented. The first of these systems is the simulation of the GSM CCCH forward error correction scheme, including both sender and receiver side implementations. The second is the jamming system used to test the various jamming methodologies tested against the simulated GSM system. The proposed jamming solutions rely on exploiting the connection-oriented property of the GSM system, as well as exploiting weaknesses in the forward error correction schemes in use. To allow the reader to gain a proper understanding of how this is achieved, each of the relevant components of the system are discussed below.

2.1 The Global System for Mobile Communication

The Global System for Mobile Communication (GSM) standards were first finalised in 1989 with the first GSM call made in Finland in 1991, from here the number of GSM connections grew massively to surpass three billion by 2008, reaching 7.7 billion in 2016 [9] [1].

2.1.1 GSM Frequency Structure

The GSM system is comprised of an uplink and downlink channel each with a bandwidth of 25MHz and separated by a 20MHz band gap. These 25MHz bands are frequency divided into 200kHz channels resulting in 124 carrier pairs [8]. Each of these 200kHz carriers are then time divided into eight time slots allowing for a total of eight users to be connected at each frequency. Each time slot duration is 0.577ms and carries 156.25 bits, this results in a bit rate of 270.833kbps [8]. The Absolute Radio Frequency Channel Number (ARFCN) is a number used to denote the frequency pair in use by the mobile. The primary GSM-900 band (P-GSM 900), which is the focus of this study, has 124 ARFCN's (1-124), this leads to uplink frequencies of 890+0.2(ARFCN) MHz, and downlink frequencies of 935+0.2(ARFCN) MHz [10]. The specifications allow for an extended GSM-900 band (E-GSM 900) which operates in the frequency ranges of 880-915MHz for the uplink and 925-960MHz for the downlink [10].

2.1.2 GSM Frame Structure

To provide the GSM system with the ability to schedule and coordinate the effective communication of various types of information that need to be transmitted between the Mobile Subscriber (MS) and the base station, various channels and frame structures are defined.

The largest GSM frame structure is a hyperframe which lasts for 3 hours 28 minutes and 53.76 seconds, this extended duration is chosen to assist with the security of the network [11] [12]. One hyperframe is composed of 2048 superframes each lasting 6.12 seconds and each superframe is then divided into 1326 frames consisting of either 26 control multi-frames, or 51 traffic multi-frames, these are created to allow for easy scheduling and synchronisation [12].

Traffic Multi-frame: The traffic multi-frame is a 26-frame multi-frame, this multi-frame has a duration of 120ms in total over the 26 transmission bursts. Of these 26 frames, 24 are dedicated to traffic alone, one is dedicated to the SACCH (discussed in *Section 2.1.3*) and one is left empty, 51 of these multi-frames exist per superframe [12].

Control Multi-frame: The control multi-frame is a 51-frame multi-frame, this multi-frame has a duration of 235.4ms in total over the 51 transmission bursts. The control multi-frame is of interest in this investigation as it is what controls the connection of mobile devices to the system, 26 of these multi-frames exist per superframe [12].

2.1.3 GSM Logical Channels

The multi-frames discussed above are then further divided into logical channels. Two categories of GSM logical channels exist, the first being traffic channels (TCH) and the second being signalling/control channels (hereafter referred to as control channels) [11].

Traffic Channels: The traffic channels are used for the transfer of speech or data traffic over the air. The GSM specifications describe two general categories of traffic channels, the first is a full rate traffic channel (TCH/F) and the second is a half rate traffic channel (TCH/H). These full rate and half rate channels can both be used either for speech or for data traffic [13].

Control Channels: The control channels are used by the network to transfer important information such as signalling or synchronisation messages to assists the mobile in utilizing the network effectively [11].

The control channels can be broken up further into three main categories, namely: Broadcast Channels (BCH), Dedicated Control Channels (DCCH) and Common Control Channels (CCCH) [14].

Broadcast Channels: The broadcast channels are responsible for supplying control channel parameters, and assisting in frequency correction and time synchronisation of the mobile device. Three broadcast channels exist, namely the Frequency Correction Channel (FCCH), the Synchronisation Channel (SCH) and the Broadcast Control Channel (BCCH) [11] [13].

Dedicated Control Channels: The dedicated control channels are used for more specific signalling to individual mobile devices for events such as call setup, authentication, control information (during calls), signal strength and handover messages [15]. Three dedicated control channels exist, namely the Standalone Dedicated Control Channel (SDCCH), the Slow Associated Control Channel (SACCH) and the Fast Associated Control Channel (FACCH) [13].

Common Control Channels: The GSM Common Control Channels can again be broken down into sub channels. On the uplink there is the Random Access Channel (RACH), and on the downlink there is the Paging Channel (PCH), Access Grant Channel (AGCH) and Notification Channel (NCH) [13]. The RACH is used by the MS on the uplink for functions such as requesting allocation of a channel for call setup, the PCH is used to page the MS for notifications such as an incoming call, the AGCH is used to notify the MS that the channel request has been granted/denied, and contains information used for further connection setup, and the NCH is used for group calls [13]. Due to the downlink signal (being received) at the MS (the receiver) being much weaker than the uplink signal (being transmitted), these downlink channels are of primary interest [16].

2.1.4 GSM Downlink Structure

The GSM downlink structure for the GSM Non-Combined channel configuration is shown in *Figure* 2.1 for timeslot 0, each letter denotes a channel which is broadcast periodically at timeslot 0 [13].

F	S	B×4	C×4	F	S	C×4	C×4	F	S	C×4	C×4	F	S	C×4	C×4	F	S	C×4	C×4	I

Figure 2.1: GSM Non-Combined Configuration Downlink Channel Structure.

Where:
F = Frequency Correction Channel (FCCH),
S = Synchronisation channel (SCH),
B = Broadcast Control Channel (BCCH),
C = Common Control Channel (CCCH),
I = Idle Time.

It can be seen that the CCCH is grouped together in groups of four, this is because the CCCH bursts are encoded and interleaved over four separate GSM bursts. The AGCH notifies the mobile device of an accepted/rejected channel request through the use of an Immediate Assignment message which contains a unique Request Reference information element to identify which mobile device the message is for [17]. Without this message and the unique reference number, the mobile device will be unable to identify the assignment is for it and thus is unable to proceed with communication. The structure for the GSM Immediate Assignment message is shown in *Table 2.1*.

Table 2.1: Immediate Assignment Message Breakdown

Information Element	**Length (bytes)**
L2 Pseudo Length	1
RR management Protocol Discriminator	½
Skip Indicator	½
Immediate Assignment Message Type	1
Page Mode	½
Dedicated mode or TBF	½
Channel Description	3
Packet Channel Description	3
Request Reference	3
Timing Advance	1
Mobile Allocation	1-9
Starting Time	3
IA Rest Octetes	0-11

The FCCH does not carry any information, but is instead a burst of all zero values to assist the MS in synchronising its frequency. Due to its all zero values this burst should be easily detectable and is used to indicate the start of the 51-Frame multi-frame. Since only one frame is sent per time slot, and the time for one frame is known to be 0.577ms, the exact times where the FCCH and hence the CCCH frames are received can be calculated.

The SCH similarly is used to assist in synchronisation of the mobile, but the SCH is used for frame synchronisation, it carries the Base Station Identity Code (BSIC) and Reduced TDMA Frame Number (RFN) [13].

The FCCH and SCH are the primary methods of frequency and time synchronisation when a mobile phone is searching for a network. The jamming of these synchronisation channels blocks a mobile from accessing the network [18]. The CCCH is used for connection-oriented operations such as call setup and SMS's, so jamming this part of the signal prevents the MS from receiving important setup parameters required to make or receive calls or SMS's.

2.2 GSM Time Diversity and Forward Error Correction Techniques

Due to GSM being a wireless communication standard, and the physical communication medium for it being air, it is prone to many forms of interference and channel losses being introduced on the physical channel. Such errors are introduced due to multiple different effects such as Doppler shifts, fading, shadowing and interference [19]. To account for this, time diversity techniques are implemented which help combat the effects of the unpredictable wireless channel.

The GSM system has multiple levels of protection to ensure the data reaches the intended recipient error free. The first of these systems is the use of a special form of block codes called cyclic codes, and a specific type of cyclic code that is constructed systematically [20] called Fire code which is known for its ability to correct single burst errors. This is not however what the code is used for in the GSM system, as this code is also good at error detection. In the GSM system the code is used exclusively for error detection and not for error correction [17] [21]. In addition to this, GSM also implements a code primarily for the use of correcting errors during transmission, this code is called convolutional code and is implemented as the next step after the Fire code [17]. Due to convolutional code being primarily used for the correction of small errors which are far apart, and errors occurring over wireless communications usually being in the form of burst errors, GSM also implements a time diversity technique known as interleaving, to ensure that if burst errors occur, they are spread out after the reversal of the interleaving process [17]. The block diagram for the GSM CCCH data flow process can be seen in *Figure 2.2*.

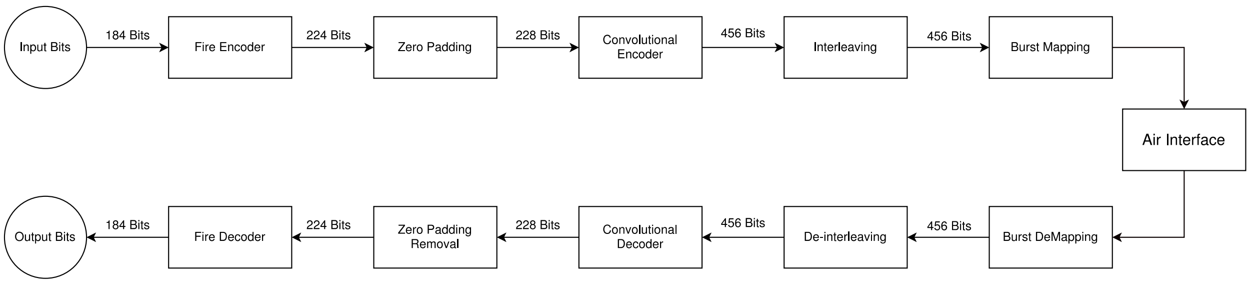

Figure 2.2: GSM CCCH Data Flow Process Diagram.

2.2.1 Fire Code

The first step and the outer code for the GSM system is a special case of cyclic burst error correcting codes called Fire codes. This code is chosen due to its strong error detection capabilities and its ease of implementation for detecting errors. Fire codes are guaranteed to detect burst errors which are much longer than their maximum error correction capabilities which makes them the ideal code for error detection on the GSM system [17].

The Fire encoder uses the systematic encoding procedure for cyclic codes, which according to [22] is summarised as follows:

Due to Fire code being a specific class of cyclic code, the same encoding procedure is followed as that for cyclic codes. To encode the message with the generator polynomial of the Fire code, first the message is shifted to allow space for the parity bits

$$u(x) = m(x)\times(x^{n-k}), \tag{2}$$

where

$$deg(g(x)) = n - k, \tag{3}$$

then divide the resulting $u(x)$ by the generator polynomial of the Fire code, storing the remainder as $d(x)$, where $d(x)$ are the parity bits. The desired encoded message is then achieved by subtracting this remainder from $u(x)$,

$$c(x) = u(x) - d(x). \tag{4}$$

Due to the initial shift, there is no overlap and the parity bits and the message bits remain separate after this subtraction.

The Fire code error detection capabilities can be summarised according to [23] as follows:

Given a Fire code defined by

$$g(x) = (x^c + 1)p(x), \tag{5}$$

where $p(x)$ is irreducible, of degree m and the two factors are relatively prime. This code will be able to detect any single burst error with length less than $c + m$, or it will be able to detect any pair of burst errors, as long as the shorter burst has a length less than or equal to m, and the length of the bursts combined is no greater than $c + 1$ bits. In the GSM system m is equal to 17, and c is equal to 23.

Therefore the GSM code is capable of detecting single bursts less than 40 bits in length, and any two bursts with a combined length less than 24 bits. Given that the combined length of the bursts is limited to 24 bits, this sets the maximum length of the shorter burst at 11 bits.

When the length of the error bursts exceeds these limitations the code is usually still able to detect errors, although not with 100% probability as there exists a case where if the received error is a multiple of the generator polynomial used to create the code, then the decoder will be unable to detect this error. This can be seen through the following decoding process which is summarised as follows:

The decoding process is very simple if error correction is not implemented and again involves division by the generator polynomial. Given the received vector as

$$r(x) = c(x) + e(x), \tag{6}$$

and since we know $c(x)$ is a multiple of $g(x)$, then dividing $r(x)$ by $g(x)$ will result in a remainder (called the syndrome) of zero if and only if $e(x)$ is either zero or a multiple of $g(x)$. Therefore, if the length of the error burst is less than that of $g(x)$ the remainder will return zero only if there is no error in the code. An addition to this case is when the received codeword $r(x)$ is all zero's, as this will also result in a remainder of zero when dividing $r(x)$ by $g(x)$.

Therefore, introducing an error which is equal to, or a multiple of the generator polynomial should prevent this code from being able to detect the errors.

2.2.2 Convolutional Code

According to [17] the GSM system comprises of both an outer and an inner code, for both error detection as well as error correction. The convolutional code is the inner code in the GSM system and is the only error correction step before the interleaving process. Before the convolutional encoding process can begin, the 224 input data bits need to be padded with four zero bits at the end as a reset for the convolutional encoder (zero termination). This convolutional code is a half rate code which is defined by the following two generator polynomials [24]:

$$G_0(d) = d^4 + d^3 + 1, \text{ and} \tag{7}$$

$$G_1(d) = d^4 + d^3 + d + 1. \tag{8}$$

A convolutional code also has a memory component, which tells it how many of the previous symbols (in this case bits) to combine with the current result. This memory component is equal to the maximum constraint length, which for the majority of GSM channels is four [17]. The 228 bits after encoding results in a block of 456 convolutionally encoded bits which are defined by the following two equations [24]:

$$c(2k) = u(k) + u(k-3) + u(k-4), \text{ and} \tag{9}$$

$$c(2k+1) = u(k) + u(k-1) + u(k-3) + u(k-4), \text{ for } k = 0,1,\dots,227, \tag{10}$$

$$u(k) = 0 \text{ for } k < 0. \tag{11}$$

The block diagram for this process is shown in *Figure 2.3*.

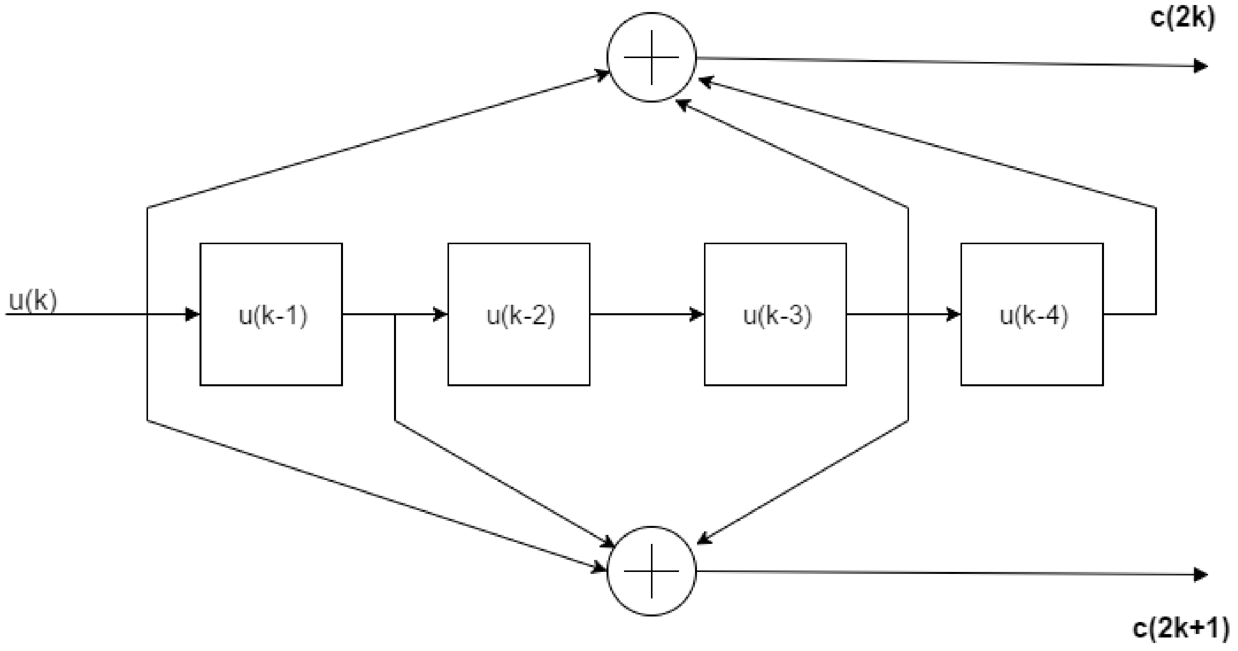

Figure 2.3: GSM Convolutional Code Block Diagram.

2.2.3 Interleaving

Interleaving, also known as bit interleaving is a process in which a message to be transmitted is first spread out over time before transmission by separating and reordering the bits from a message before transmission, and then re-organising the messages into their original form at the receiving end. The interleaving process is done to reduce the amount of damage burst errors cause to transmitted messages, so that error correcting codes such as the convolutional code used in the GSM system has a higher chance of being able to correct the errors. This is due to convolutional code being good at correcting errors as long as they are small and spaced far enough apart.

There are two different types of interleaving in use in the GSM system each having its own benefits. The first type of interleaving to be discussed is rectangular interleaving. The rectangular interleaving process requires that the sender waits until a full block of 456 bits is ready to be sent, it then reads the data out in a different predefined order, dividing the data up into four sections (this is known as the interleaving depth) of 114 bits. The data is again divided into 8 sub blocks by placing the even numbered bits in the first half of each CCCH timeslot (bit 0-57), and the odd numbered bits in the second half of each CCCH timeslot, this is known as burst mapping. The result of the interleaving and bit organisation into 8 sub blocks is shown in Figure 2.4. This type of interleaving is used for the control channels in the GSM system, the bits are reordered according to the following formula [24]:

$$i(B,j) = c(n,k) \quad \text{for} \quad k = 0,1,\ldots,455, \text{ and} \tag{12}$$

$$n = 0,1,\ldots,N,N+1,\ldots,$$

$$\text{where } B = B_0 + 4n + k \, mod \, 4, \text{ and} \tag{13}$$

$$j = 2\big((49k) mod\ 57\big) + \left(\frac{k \, mod \, 8}{4}\right). \tag{14}$$

The second type of interleaving in use in the GSM system is known as diagonal interleaving, in this interleaving process the data for one block is divided again into sub blocks, but unlike in rectangular interleaving, the blocks of 456 bits are not kept within that block, but now each block is merged with the block before and after it. Diagonal interleaving is used mostly for speech channels [17]. Due to the CCCH using rectangular interleaving, diagonal interleaving is not discussed in further detail.

CCCH Block 1		CCCH Block 2		CCCH Block 3		CCCH Block 4	
Bits 0-56	Bits 57-113	Bits 0-56	Bits 57-113	Bits 0-56	Bits 57-113	Bits 0-56	Bits 57-113
0	228	57	285	114	342	171	399
64	292	121	349	178	406	235	7
128	356	185	413	242	14	299	71
192	420	249	21	306	78	363	135
256	28	313	85	370	142	427	199
320	92	377	149	434	206	35	263
384	156	441	213	42	270	99	327
448	220	49	277	106	334	163	391
56	284	113	341	170	398	227	455
120	348	177	405	234	6	291	63
184	412	241	13	298	70	355	127
248	20	305	77	362	134	419	191
312	84	369	141	426	198	27	255
376	148	433	205	34	262	91	319
440	212	41	269	98	326	155	383
48	276	105	333	162	390	219	447
112	340	169	397	226	454	283	55
176	404	233	5	290	62	347	119
240	12	297	69	354	126	411	183
304	76	361	133	418	190	19	247
368	140	425	197	26	254	83	311
432	204	33	261	90	318	147	375
40	268	97	325	154	382	211	439
104	332	161	389	218	446	275	47
168	396	225	453	282	54	339	111
232	4	289	61	346	118	403	175
296	68	353	125	410	182	11	239
360	132	417	189	18	246	75	303
424	196	25	253	82	310	139	367
32	260	89	317	146	374	203	431
96	324	153	381	210	438	267	39
160	388	217	445	274	46	331	103
224	452	281	53	338	110	395	167
288	60	345	117	402	174	3	231
352	124	409	181	10	238	67	295
416	188	17	245	74	302	131	359
24	252	81	309	138	366	195	423
88	316	145	373	202	430	259	31
152	380	209	437	266	38	323	95
216	444	273	45	330	102	387	159
280	52	337	109	394	166	451	223
344	116	401	173	2	230	59	287
408	180	9	237	66	294	123	351
16	244	73	301	130	358	187	415
80	308	137	365	194	422	251	23
144	372	201	429	258	30	315	87
208	436	265	37	322	94	379	151
272	44	329	101	386	158	443	215
336	108	393	165	450	222	51	279
400	172	1	229	58	286	115	343
8	236	65	293	122	350	179	407
72	300	129	357	186	414	243	15
136	364	193	421	250	22	307	79
200	428	257	29	314	86	371	143
264	36	321	93	378	150	435	207
328	100	385	157	442	214	43	271
392	164	449	221	50	278	107	335

Figure 2.4: Interleaving and reordering process for each GSM CCCH timeslot.

2.3 Signal Jammers

A signal jammer is a device used to disrupt, partially or completely, the effective radio communication between one or many entities on a communication channel. In wireless communication this is done by transmitting a disruptive signal on the same frequency as that being used for communication, effectively drowning out the original messages and causing a decoding failure on the receiving end [25]. The disruptive signal is usually generated from a random noise source and should be transmitted at a high enough power to prevent original signal from being decoded correctly.

2.3.1 Current Signal Jammer Implementations

Due to signal jamming being a widespread topic of interest, there have been many different forms of jamming attack models discovered, reviewed and tested. According to [3], [26] and [27] there are four common effective categories of jammers that currently exist, these are: Constant Jammers, Deceptive Jammers, Random Jammers and Reactive Jammers. These are discussed below with the advantages and disadvantages of each method given.

Constant Jammers: Constant jammers are the most basic form of signal jammers that exist, they work on an always on basis which means that they are always broadcasting regardless of the state of the channel they are attempting to jam. When broadcasting they transmit random noise on the channel. This causes one of two effects on the channel depending on the nature of the system being jammed, either the system will continue to transmit and have its signal drowned out by that of the interfering signal, or if it's a channel sensing system relying on threshold signal strengths to determine channel availability, it will continuously see the channel as being busy and never find a chance to transmit.

Deceptive Jammers: Deceptive jammers work similarly to constant jammers, although they have a subtle yet definitive difference. They also work on an always on basis, but instead of transmitting random noise, they transmit legitimate packets, which tricks the network into thinking legitimate communications are taking place on the channel making it difficult for legitimate communications to take place (because the channel always appears busy). This makes it harder to detect that a jamming attack is being executed, but still requires large amounts of energy due to its always on status.

Random Jammers: Random jamming is a more energy efficient method of jamming, although it is not always as effective as constant jammers. In the random jammer methodology, the signal jammer alternates between periods of transmitting jamming signals and sleeping (where no jamming signal is transmitted). The transmission times and sleep times are chosen during jammer setup and through this the jammer can be made more energy efficient by reducing jamming time. The downfall to this method of jamming is that the jammer is not aware of the channel usage and thus may sleep during times where jamming is required, or may transmit a jamming signal when the channel is quiet and there is no legitimate signal to be jammed.

Reactive Jammers: In the three jamming strategies mentioned above there exists a common property which requires the jammers to be transmitting for the majority if not all of the time. This property is that the above jammers do not take into account the communication patterns occurring on the channel, and while effective, these jammers are also very inefficient when it comes to energy consumption [28]. Another approach to jamming exists in which the jammer is aware of the channel it is trying to jam, in this jamming methodology the jammer first listens to the channel on which communications are to take place, and then upon detection of a legitimate communication on the channel, the jammer starts to transmit a jamming waveform. This method of jamming is known as reactive jamming as the jammer only transmits in reaction to detecting communication on the channel. This system can however still use a lot of energy in situations where there is almost constant legitimate communication on the channel.

2.3.2 Advanced Signal Jammer Implementations

From information theory it can be seen that for a packet encoded with Forward Error Correction (FEC) to be decoded correctly, it does not require for the entire packet to be received correctly, a limit exists as to how much of the packet can be corrupt before the original packet can no longer be recovered. From the perspective of signal jamming it can be said that the entire packet does not need to be jammed to stop it from being successfully decoded at the receiver. Having knowledge of the error correction schemes in use allows for this property to be exploited such that the jamming signal can be active for a lesser time and thus have an overall lower power consumption.

Having knowledge and a good understanding of the protocols in use in a communication system is another advantage when attempting to optimise jamming strategies. The reason for this is that often in a communication system a substantial part of the information being transferred is overhead to set up and maintain the connection, this includes parameters such as timing, number of users and transmission power levels [29]. As a result, only a part of this information is actually used for active communication with other users. A GSM mobile network is a good example of this, as when a call, SMS or data transfer is not taking place, the system is just providing connection parameters keeping the mobile phone connected to the network, and keeping both the network and phone up to date with important information such as the available operator towers and the current location of the mobile (for call routing) [29]. From this it can be seen that if the goal is to prevent phone calls from being made or received, an always-on jammer is unnecessarily energy heavy, and even through the use of reactive jamming methods the jammer will still be jamming a lot of unnecessary packets which also leads to increased power requirements and an easier to detect jammer. It is by making the jamming system intelligent and protocol aware, that one can target only those packets involved in the connection setup procedure and thus eliminate unnecessary jamming transmissions and hence reduce required transmission times while maintaining successful jamming.

2.3.3 Intelligent Signal Jamming

In *Section 2.3.1* it is seen that signal jammers can be implemented in a variety of ways depending on the requirements of the system. To be effective on a communication channel where there is a lot of traffic, all four jamming methodologies discussed above share the common property of having large energy requirements due to lengthy jammer transmission times. A proposed solution to this is making the signal jammer intelligent by making it aware of the protocol being used by the communication system. In [27] it is proposed that by taking into account the error correction capabilities of protocol being used, it is possible to make the jammer more efficient by only interfering with the minimum required portion of the packet to cause decoding failure and hence cause the packet to be discarded at the receiving end.

Intelligent signal jamming is a newer jamming strategy aimed at improving on the overall energy efficiency and decreasing the detectability of current jamming systems. It takes the concept of a protocol aware jammer to the next step by targeting specific control packets and crucial timings, an example of this can be seen in [6] for 802.11b Networks. For this research we propose an intelligent jamming strategy which takes advantage of the connection-oriented property of the GSM protocol by attempting to only target the crucial packets required for connection setup, and thus prevent communications by only jamming a fraction of the packets being transmitted. Another property that this provides is that the jamming signal becomes harder to detect due to its intermittent nature [4].

2.3.4 Signal Jammer Synchronisation with the GSM System

For a signal jammer to be able to utilise the intelligent and advanced jamming strategies discussed above, it is of utmost importance that the jammer is able to synchronise its system with the original system being jammed. This will ensure that the jammer is able to transmit at the specific time required to interfere with the original message effectively. The synchronisation procedure is briefly discussed below for completeness, but its implementation is beyond the scope of this research.

Frequency Synchronisation: The first step in the synchronisation process is finding the correct frequencies on which the jammer needs to transmit. In order for the jamming system to synchronise in frequency with the GSM systems original transmitter, the jammer will first need to find the beacon frequencies in use by the system, these are frequencies on which the system continuously transmits the Control Multi-frame (discussed in *Section 2.1.2*). Once the jammer is aware of the beacon frequencies in use by the original transmitter it can move on to time synchronisation.

Time Synchronisation: The next step in the synchronisation process is finding the correct times on these frequencies to transmit the jamming signal. The time synchronisation process is discussed considering only a single frequency, the time synchronisation process can be repeated for each additional frequency found. The GSM system is divided into 8 timeslots, the first of these timeslots, denoted "timeslot 0" is usually used for the Control Multi-frame, with the other seven timeslots being used to transmit traffic information. The jammer is able to find timeslot 0 by searching for the FCCH (discussed in *Section 2.1.4*) and calculate the position of the relevant channel to jam based on the known GSM transmission durations and equation (1). The transmitter would then have to calculate the required transmission time based on distances from the original transmitter and the original receiver.

2.3.5 Related Research

In this section similar works are discussed which were the inspiration for this research. Research involving the energy efficiency of jamming systems has been investigated in the past for a variety of different signal protocols. A variety of different methods have been used to achieve these energy efficient systems.

In [30] Nguyen et al. proposed an energy efficient reactive jamming system for WiFi (802.11g) and mobile WiMAX (802.16e) networks. The reactive jammer was designed with the choice of different detection techniques such as high or low energy detection (to indicate the start of transmission) as well as a cross correlation function to correlate with known packet preambles (to ensure that it is legitimate packets being detected). Once a legitimate packet is detected, the transmission chain is initialised, and a jamming waveform is transmitted. Three types of jamming waveforms are configured on the system, the first being a pseudorandom white Gaussian noise signal at 25MHz, the second being a repetitive replay of previously received samples, and the third being a chosen waveform which can be streamed from the host machine. The reactive jamming system was implemented on the FPGA on a USRP N210 to allow for a fast system response time, and interfaced with a GNU Radio Companion application on the host machine. With 0.1ms jamming duration on each detected packet, no packets were correctly received at the receiver when the Signal to Noise Ratio (SNR) dropped below 15.94dB. For a jamming duration of 0.01ms on each packet, no packets were correctly received after the SNR dropped below 2.79dB at the receiver. The system was tested using a network bandwidth measurement tool called iperf.

In [31] Acharya et al. also investigated energy efficient jamming methods for WiFi networks and the focus of the investigation was on the 802.11b wireless network standard. Various different jamming approaches are investigated and tested on a network simulation software package called OPNET 10.0. There are three main categories of jamming methodologies which are investigated: 1) *Trivial Jamming*, which is what is referred to in this paper as *Constant Jamming,* wherein the jammer is always transmitting an interfering waveform. 2) *Simple Periodic Jamming*, which is referred to in this paper as *Random Jamming,* in which the jammer transmits periodically for a predefined period of time, and then stays quiet for a predefined period of time. 3) *Intelligent Jamming*, in which the jammer is made protocol aware, and focusses on jamming specific control or data packets to effectively stop network throughput. These three jamming methodologies are compared according to their energy usage and the results are as follows: it is shown that for Constant Jamming all communications are blocked although this is at a high energy cost. For Periodic Jamming it is shown that as long as the period between successive jamming pulses is small enough, it can block all communication with an improved energy efficiency of three to four times more than Constant Jamming, if the silence period is too long, some packets may slip through without being corrupted. For Intelligent Jamming it is shown that energy efficiency improves by a factor of up to five times, while the system is capable of blocking all communication on the channel.

In [4] Wilhelm et al. investigate the ZigBee 802.15.4 wireless protocol's resilience against reactive jamming techniques. The jamming methodology focusses on attacking the physical layer. The hardware used for this research was a USRP2 using the on-board FPGA and two MICAz motes. Various causes of loss on the physical layer are analysed and the theoretical influences of jamming each part is discussed, these include symbol misdetection in which the Cyclic Redundancy Check (CRC) check fails from jamming individual symbols, failed timing caused from interfering with preambles, frame synchronisation from corrupting packet length fields, and corrupting the Automatic Gain Control (AGC) to create clipping or cause very low transmit power. All of these cause the receiver to be unable to detect packets correctly. Three different jamming waveforms are investigated, these are Wideband Noise, Narrowband Noise and legitimate packets with varying loads. The effectiveness of wideband noise is limited in this experiment due to the limited transmission power of the USRP2. Single tone jamming proved the most effective as the narrowband power caused the AGC to adjust quickly thus causing packet loss, another cause is the jamming signal is detected as a second carrier frequency resulting in the phase correlation process failing. Jamming with modulated signals did not perform as well as the previous two waveforms, requiring the jammer gain to be three to four dB higher before the throughput was reduced to zero. In this research they managed to get a reaction time as little as 20µs.

In [18] Petrecca et al. similar control channel jamming is done to that in this research. In the paper the effectiveness of jamming the Broadcast Control Channel (BCCH), the Frequency Correction Channel (FCCH) and the Synchronisation Channel (SCH) is tested. The simulations are done using a PHY layer simulator developed in MATLAB. All of these jamming strategies are targeted against the mobile device's synchronisation ability with the base station, due to this all jamming strategies require an initial 5.296s burst to cause coarse synchronisation on the channel, and after this the individual control channel jamming begins. The FCCH jamming procedure blocked all communications requiring a

SNR not greater than 6dB at the receiver, for the SCH jamming the effective jamming is achieved with a SNR no greater than -9dB at the receiver. The BCCH jamming procedure blocked all communications requiring a SNR not greater than -6dB.

2.4 Conclusion

In this chapter the relevant literature required to complete this research is provided, aiding the reader in properly understanding the work presented. In *Section 2.1* the GSM system is introduced and explained, in *Section 2.2* the GSM channel coding schemes are presented, focussing on the CCCH. In *Section 2.3* various basic jammer implementations are explored, discussions into what improvements have been made to these implementations, and recent similar works are discussed. In *Section 3.1* the investigative approach to this research is provided, with *Section 3.2* giving details on each of the jamming methodologies investigated.

Chapter 3: Research Methodology

This chapter describes in detail the methodology undertaken to complete the research, each of the jamming procedures tested is discussed individually with the implementation details of each provided.

3.1 Investigative Approach

In this research we set out to answer the research question:

"How can we exploit the control channel forward error correction scheme of the GSM system in order to minimize the number of jamming bits required to prevent communications on the channel?"

To answer this question, a simulated testing environment is created in which the proposed jamming methodologies are tested. Each of the key components involved in the simulation of this GSM control channel testing environment is presented in this section.

3.1.1 Experimental Setup

To implement both the GSM CCCH forward error correction scheme and the different jamming methodologies, the MATLAB simulation software is used on a Windows desktop environment. In the simulations the forward error correction and time diversity techniques present in the GSM system common control channel (CCCH) are implemented.

Due to this investigation being done on a bit level, the burst mapping procedure (see *Figure 2.2*) is excluded from the simulation as it will have no effect on the outcome of the research.

The final simulated system setup is shown in *Figure 3.1.*

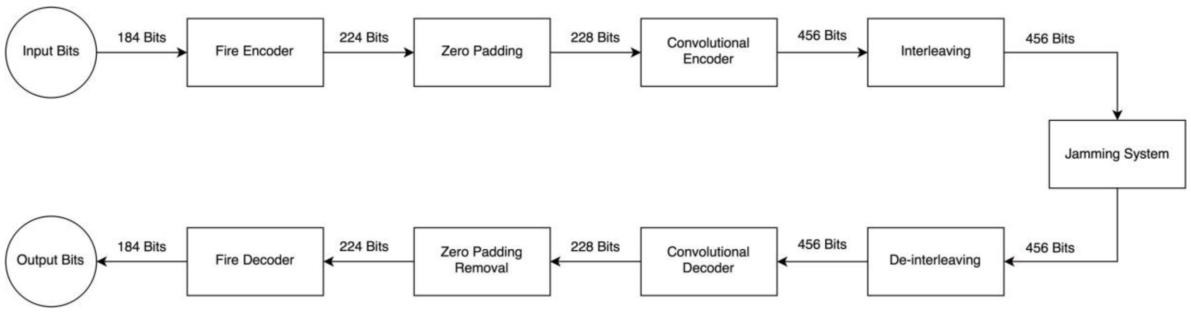

Figure 3.1: Simulated system data flow diagram

3.1.2 Convolutional Encoder and Decoder

The convolutional code used in the GSM CCCH system is a ½ rate code with constraint length five. This code requires 228 input bits and results in 456 convolutionally encoded bits at the output of the encoder, each made up of a combination of the current input bit and up to four previous input bits. To implement the convolutional encoder in MATLAB, the inbuilt convolutional encoder from the Communications System toolbox is used [32]. The convolutional encoder takes in the 228 bits from the Fire encoder and convolutionally encodes them according to the following two generator polynomials [24]:

$$G_0(d) = d^4 + d^3 + 1, \tag{15}$$

$$G_1(d) = d^4 + d^3 + d + 1. \tag{16}$$

This process is illustrated in *Figure 3.2*.

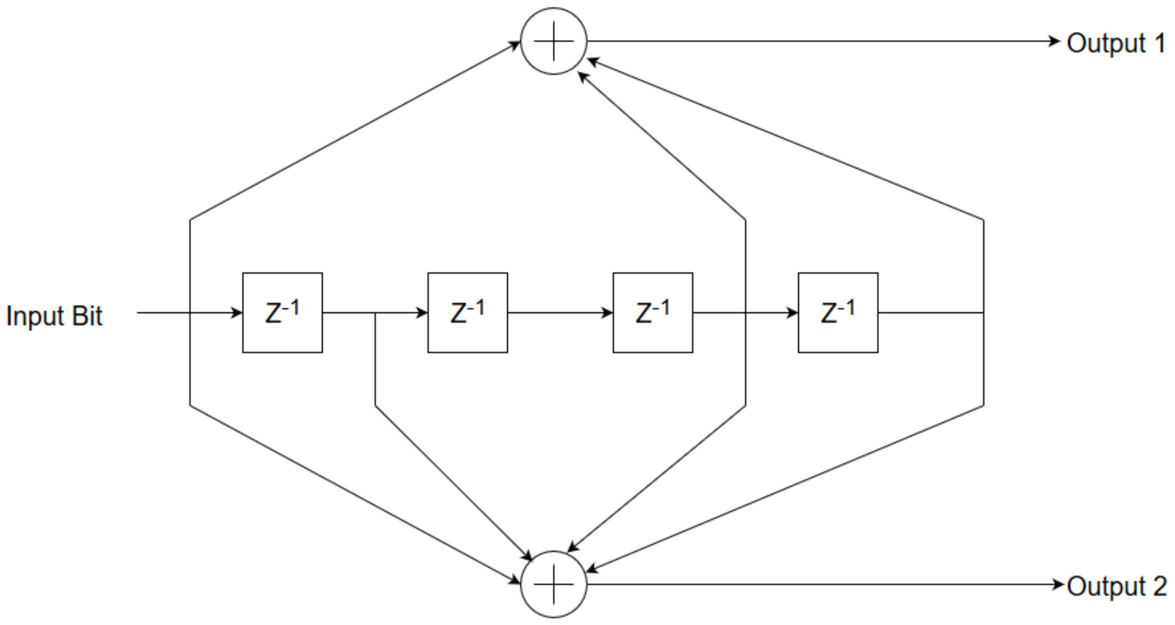

Figure 3.2: ½ Rate convolutional encoder block diagram with constraint length 5.

The decoder used is a Viterbi decoder which is also part of the Communications System toolbox in MATLAB, it is configured with a traceback length of 25, soft decoding, zero termination, and the same trellis structure as used for encoding. These parameters are chosen in accordance with those used in the GSM CCCH to ensure the decoder performance matches that of the real system.

3.1.3 Fire Encoder and Decoder

The outer code used in the GSM forward error correction scheme is Fire code, this code is used primarily for the purpose of error detection in the GSM system, and as such allows the system to detect when there are errors which were not corrected by the convolutional decoding process. In the GSM system this Fire code is used for most of the signalling channels and it is constructed from the following generator polynomial:

$$G(x) = (x^{23} + 1)(x^{17} + x^3 + 1). \tag{17}$$

The Fire encoding and decoding procedures are outlined in *Section 2.2.1.*

To implement the Fire encoder and decoder in MATLAB, no existing Communications System toolbox functions were available and as such the Fire encoder and decoder were implemented manually. The encoder works by taking 184 data bits (one block) and adding 40 parity bits to create a total of 224 encoded bits. This is done in a systematic manner which results in the data and the parity bits being separate parts of the encoded message. In the GSM system this is done in such a way that the parity bits are first followed by the data bits, resulting in 224 encoded bits being in the form of [parity:data] as shown in *Figure 3.3.*

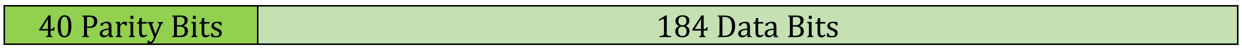

Figure 3.3: Fire code systematic encoding structure.

3.1.4 Jamming System

To test the GSM CCCH forward error correction scheme, four jamming strategies are implemented. These strategies are aimed at improving on current jamming methodologies by combining the works done in [27] and [18]. In [18] Petracca et al. attempted to improve on existing GSM jamming strategies by targeting only crucial synchronisation and control data. In [27] Hussain et al. proposed that by taking into account the error correction capabilities on a channel, the required jamming duration can be reduced while still causing decoding failure at the receiver. In the GSM system, the CCCH is responsible for transmitting the crucial setup information required for completing connection oriented processes such as calls and SMS's amd it's forward error correction scheme uses the convolutional and Fire code discussed above.

Due to these methodologies targeting only the CCCH, and not the synchronisation data, the mobile device should remain connected to the network with no signs of it being jammed until the user attempts to make a call or SMS. However this is not the aim of this research and is mentioned for the interest of the reader.

Each of these jamming strategies is focussed on exploiting different parts of the GSM CCCH forward error correction scheme, the GSM system is tested as a complete system as in practice. The jamming system is modified to accommodate this by processing each of the jamming patterns before transmission. This allows for a more accurate representation of a real life jammer, and for a fair comparison to be made between each of the jamming strategies as they are all tested against the same system. To achieve this, each error pattern requires a different level of processing before being introduced onto the channel, this is discussed individually for each method. Each of the jamming strategies tested is briefly discussed below showing block diagrams of the systems implemented.

The first jamming strategy implemented is a random jamming approach, in which the CCCH FEC scheme is tested with reference to the number of bits that are jammed in a packet, irrespective of the location of these jammed bits. This is the most basic jamming approach tested, exploiting only that the FEC scheme has error correction limits (discussed in *Section 2.3.2*). In this experiment the errors are introduced after the interleaving process as they would be on the air, and as such requires no additional processing before transmission.

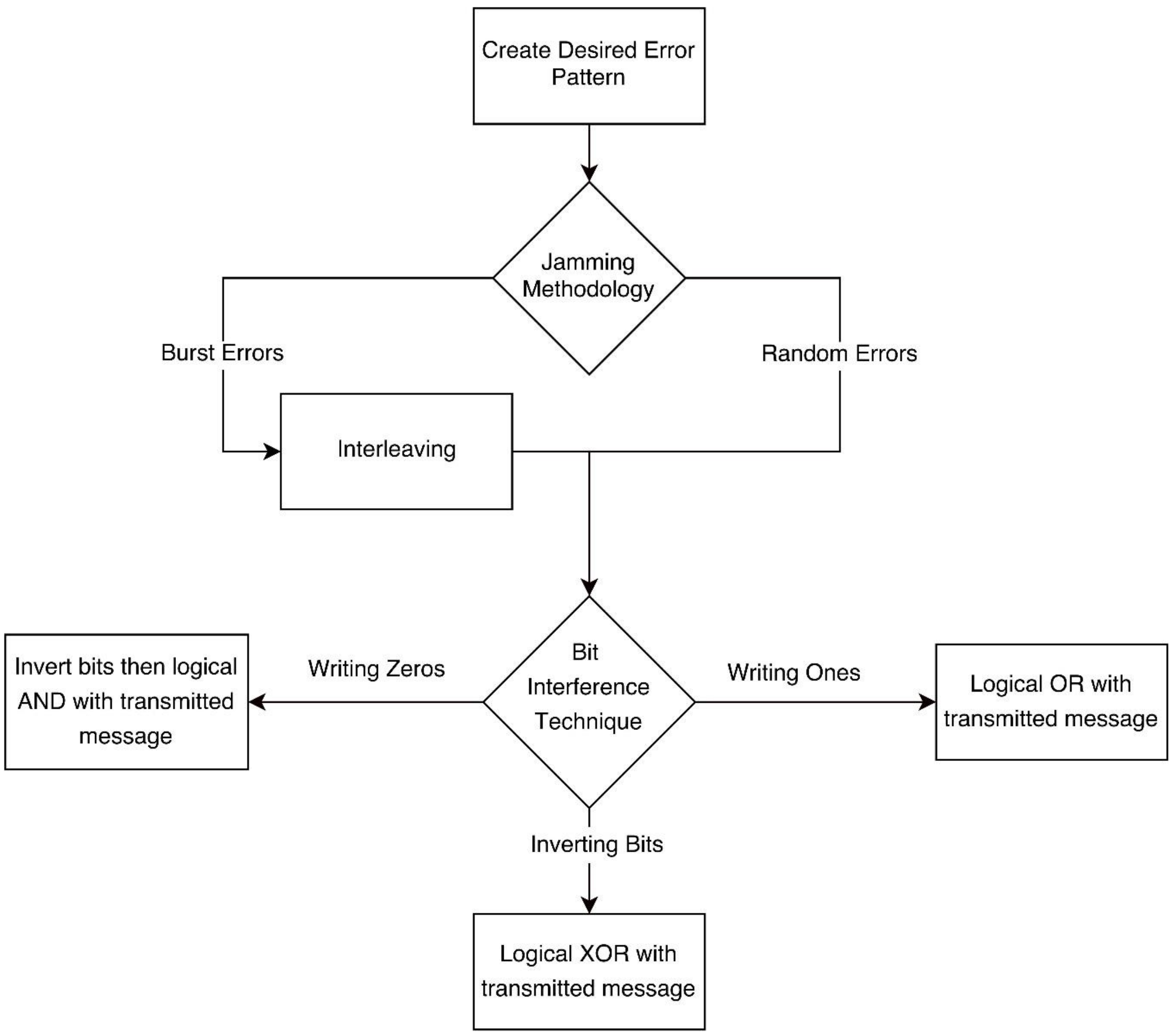
Create Desired Error Pattern
Jamming Methodology
Burst Errors
Random Errors
Interleaving
Bit Interference Technique
Writing Zeros
Writing Ones
Inverting Bits
Invert bits then logical AND with transmitted message
Logical OR with transmitted message
Logical XOR with transmitted message

Figure 3.4: Random and burst error jamming system data flow diagram.

The second jamming strategy implemented is a burst error approach, which is aimed at targeting the weak error correction capability of the convolutional code when dealing with lengthy error bursts. To ensure these bursts reach the convolutional code in complete bursts as intended, the desired burst pattern is first interleaved with the same interleaving procedure as that used in the GSM system. This ensures the correct bits are corrupted on the air, and results in the original jamming pattern being recovered after the deinterleaving procedure.

The block diagram for the jamming system implemented to test the random error and burst error jamming techniques is shown in *Figure 3.4.*

The following two methodologies are designed with bit patterns targeted against the Fire code error detection procedure. For these error patterns to reach the Fire decoder unmodified, the error patterns are passed through the same convolutional encoder and interleaver used in the GSM CCCH FEC scheme before being transmitted by the jammer. Due to convolutional code being a linear block code it shares the linear property of block codes in which the linear combination of any two valid codewords results in a third valid codeword [20]. Therefore if we encode our error pattern with the same convolutional encoder as that used in the GSM system, when we combine it with the transmitted message by means of the XOR procedure, the resulting message at the receiver will also be a valid codeword. Due to this, and the interleaver ensuring the correct bits are corrupt, the convolutional decoder does not detect or attempt to correct these errors, and they are decoded back to their original form before reaching the Fire decoder.

The third jamming strategy is designed with the following knowledge (discussed in *Section 2.2.1*) of the GSM Fire decoder in mind: Firstly, the GSM Fire code is a strong error detection code, capable of detecting all single burst errors under 40 bits in length. The second is that the GSM Fire code is used only for error detection, and if an error is detected by the Fire decoder at the receiver, the packet is discarded. This experiment is aimed at finding an error pattern in the Fire code, which when convolutionally encoded, minimises the number of bits requiring corruption by the jammer.

The fourth jamming strategy implemented is a custom jamming pattern aimed at targeting a weakness in the Fire code error detection procedure. Due to the Fire code error detection procedure involving a division by the generator polynomial used to define that Fire code, if we create an error pattern in the transmitted message which mimics the action of the original message being XOR'd with the generator polynomial (after decoding), the fire code error detection procedure should be unable to detect this as an error in the message (the reason for this is discussed in *Section 2.2.1*). This experiment is aimed at minimising the number of jamming bits required to cease communication, but sacrifices the extent to which the number of jamming bits required is reduced, to provide the added benefit of the receiver not detecting the presence of the errors.

The proposed jamming system data flow diagram for the third and fourth jamming methodologies is shown in *Figure 3.5.*

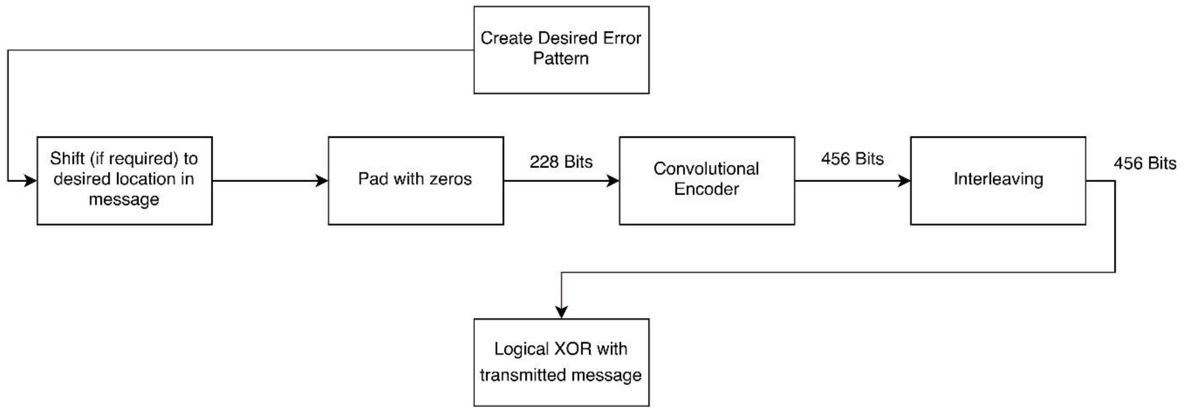
Create Desired Error Pattern
Shift (if required) to desired location in message
Pad with zeros
228 Bits
Convolutional Encoder
456 Bits
Interleaving
456 Bits
Logical XOR with transmitted message

Figure 3.5: Proposed jamming system data flow diagram.

3.2 Jamming Methodologies

There are three different bit corruption approaches considered when testing the jamming methodologies discussed below. Not all bit corruption approaches worked for all jamming methodologies, but these are discussed individually. In this section a numerical '1' or '0' is used when referring to a binary bit value.

3.2.1 Bit Corruption Techniques

Inverting Bits: Due to the nature of GMSK modulation, it is possible to replace specific bits in the air with other ones by simply transmitting the desired bit at a higher power and at the same time as the original undesired bit is transmitted. To be able to invert the bits on the air however, a device is required which has a very fast processing speed which can detect the original bit being transmitted and transmit the opposite (binary inverse) bit in time for the existing bit to be overwritten. To simulate the bit inversion in software, the error locations are marked with 1s in an otherwise zero array, and the binary XOR function is used to introduce the errors into the message. The bit modification diagram for this method can be seen in *Figure 3.6.*

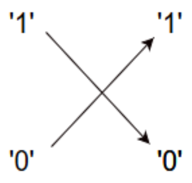

Figure 3.6: Bit modification diagram when inverting bits.

Writing 1s: For this part of the testing procedure, the error patterns are introduced by replacing specific bits at specific locations with 1s. This type of jamming is realisable with much cheaper hardware as it does not require the same level of processing speeds as inverting the bits would require. This is due to the fact that it does not need to know what the original bit is to overwrite it with a 1. To simulate replacing bits with 1s in software, the error locations are marked with 1s in an otherwise zero array, and the binary OR function is used to introduce the errors into the message. The bit modification diagram for this method can be seen in *Figure 3.7*.

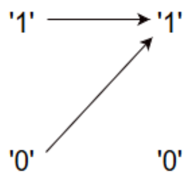

Figure 3.7: Bit modification diagram when writing 1s.

Writing 0s: Due to the fact that the information which is sent over the air in GSM is a combination of modulated binary bits, the alternative to the low cost jamming method of replacing bits with 1s is replacing bits with 0s. Again, because this doesn't require that the jammer knows what the original bit being transmitted is, it is able to overwrite the bits with 0s without the use of expensive processing hardware. To simulate replacing bits with 0s in software, the error locations are marked with 0s (with all other bits being 1), and the binary AND function is used to introduce the errors into the message. The bit modification diagram for this method can be seen in *Figure 3.8*.

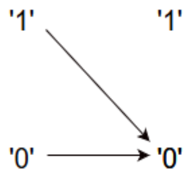

Figure 3.8: Bit modification diagram when writing 0s.

The above error corruption techniques are used when testing the four jamming methodologies mentioned in *Section 3.1.4.* An in-depth discussion on their implementations is provided in *Section 3.2.3*.

To assist the reader in understanding each of the jamming implementations to follow, an example of a bit error propagating through the proposed system (shown in *Figure 3.5*) is provided in *Section 3.2.2*. The proposed system is chosen for this example as it involves all possible steps for the other jamming methodologies. In the sections to follow, the terms "transmitted" is used to refer to the original legitimate message being transmitted and the term "introduced" is used to represent one of the error messages being applied to the original message through one of the bit corruption techniques above to result in the received corrupted message. This process can be described as follows:

$$c_r(x) = c_t(x) +\_ \, e_t(x), \tag{18}$$

where $+\_$ represents the addition through one of the bit corruption techniques, $c_r(x)$ is the received corrupted message, $c_t(x)$ represents the error free message before transmission on the channel and $e_t(x)$ represents the final error message to be introduced. The received message $c_r(x)$ is then decoded as it would be on the GSM system (see *Figure 3.1*) resulting in the 184 bit message $m_r(x)$ at the receiver.

3.2.2 Example of an Error Pattern Propagating Through the Proposed Jamming System

Step 1: Creating the desired jamming pattern. This depends on the required effect at the receiver, for the purpose of this example we will consider only one bit placed at bit position one in the error message. The maximum length of this jamming pattern is equal to the data length of 184 bits.

Step 2: The shifting of the message is only required for the fourth jamming strategy in which the errors introduces are not detected, to ensure the errors occur at the desired bit locations. The errors will pass through the convolutional and Fire decoder undetected, and the location of the errors in the error message will corrupt those same bit locations in the received message after the decoding process. This shift is achieved by prepending

0s to the error message. When choosing a desired error location, it is important to remember that the Fire encoder results in 224 systematically encoded bits of which the first 40 are parity bits.

Step 3: Pad with 0s to reach a total of 228 bits, the last four 0s serve as a terminating sequence for the convolutional encoder. The error pattern at this step of the process is the required input for the convolutional encoder and is denoted as $e_c(x)$.

Step 4: The 228 bit $e_c(x)$ is fed into the convolutional encoder, the output of this is a 456 bit error pattern. The error pattern at this step of the process is the required input for the interleaving process and is denoted as $e_i(x)$. Let it be noted that this is the point in the process where the burst error jamming approach starts, as such the desired 456 bit $e_i(x)$ is created here and is not taken through any of the previous encoding steps for this method.

Step 5: The 456 bits $e_i(x)$ is the second last step before the errors are introduced onto the channel. To ensure the correct bits are corrupted on the air, $e_i(x)$ is fed through the same interleaving process as that used in the GSM CCCH. The output of this step is a 456 bit interleaved error pattern. The error pattern at this step of the encoding process is the final message before being introduced or "transmitted" on the channel and is denoted as $e_t(x)$.

Step 6: The introduction of $e_t(x)$ on the channel involves one of the three bit-corruption techniques being applied to the original encoded message $c_t(x)$, resulting in the corrupted message $c_r(x)$ at the receiver. Let it be noted that this is the point in the process where the random error jamming approach starts, and as such the 456 bit $e_t(x)$ is created here and is not taken through any of the previous encoding steps.

The above steps (or a subset of which) are used to process the error patterns for each jamming methodology before being introduced. Each of the jamming methodology implementations are described in detail below using the same symbols as above to denote the error messages at each stage of the process.

3.2.3 Random Error Locations

For this experiment, the errors are introduced after the interleaving process to simulate random errors occurring on the channel. Due to the way in which this is done after the Fire encoding step, there is no guarantee that the output from the convolutional decoder won't have errors in the parity check section of the Fire codeword. Each experiment runs for error counts of one up until 456 errors are introduced, and a total of 10000 random data streams are tested for each error count. The averages over the 10000 different data streams are then taken as the final results for that error count.

Inverting Bits: For this implementation, a program was created in which the error array $e_t(x)$ is originally filled with zero values, and then for each iteration, going from one up to the maximum number of errors (456), errors are introduced at random locations by placing 1s in $e_t(x)$ where the error is to be introduced into $c_t(x)$. For each iteration it is enforced that the number of errors introduced is equal to the iteration number, for example if a location is chosen that already includes an error, a new random location is chosen until another error can be introduced. This error pattern, $e_t(x)$, is then XOR'd with the message, $c_t(x)$, simulating the bit corruption on the air, and resulting in the received error message $c_r(x)$.

Writing 1s: For this implementation the goal is to test the effectiveness of introducing 1s at random locations in the original transmitted message $c_t(x)$. To introduce 1s into $c_t(x)$, the same procedure is followed as that for inverting bits but instead of inverting each bit by means of the XOR command, 1s are introduced by using the logical OR command to OR the message $c_t(x)$ with the interleaved error message $e_t(x)$. Again this is repeated for 10000 iterations for each error count, again from one up until 456 errors. It is notable that the number of errors in the received message $c_r(x)$ is not equal to the number of 1s in the introduced error message $e_t(x)$ as some 1s are written where a 1 already exists in $c_t(x)$.

Writing 0s: For this implementation, a similar procedure is followed as with the writing 1s simulation, but because we are writing 0s at specific locations, the error array $e_t(x)$ is initially filled with 1s, and then 0s are placed at random locations where errors are to be introduced, also starting with one and increasing up till 456. This is done in this way because when introducing 0s the AND operator is used, so all the 1s keep the existing bits the same, while all the 0s write 0s at those locations. It is notable that in this implementation the same situation occurs as that in the writing 1s simulation where the number of corrupt bits in $c_r(x)$ is no longer equal to the number of 0s written by $e_t(x)$, because we are writing specific bits now (0s), there is a probability that the position where the 0 is written is in a position where a 0 already exists and thus no change is made to the transmitted message $c_t(x)$. As this is a binary bit stream generated with random values, this probability is approximately 50% for each zero introduced that it does not corrupt a bit.

3.2.4 Errors in Bursts

In this simulation the convolutional decoder is tested against burst errors. The burst errors are introduced before the interleaving process to ensure they return to a burst after being deinterleaved by the receiver. The errors are introduced in multiple burst sizes, starting with a burst size of one and increasing until a burst size of 40 (depending on the simulation). The experiment is repeated for each burst size with the location of each burst being varied on each iteration, the bursts start at bit zero, and are shifted through each possible location in $e_i(x)$ by increasing the bit number on which the burst starts by one.

Inverting Bits: For this implementation the errors are introduced in a burst fashion. For this simulation in which the bits are inverted on the air, it follows the same procedure as before in which the error array $e_i(x)$ is marked with consecutive 1s (depending on burst size) in the desired locations then $e_i(x)$ is interleaved resulting in the error message $e_t(x)$ with errors in the appropriate locations where they are to be introduced into $c_t(x)$, in this case where the bits are to be inverted. The errors are introduced into the transmitted message by means of the logical XOR operation. This test is completed for a variety of different burst lengths ranging from one up to seven and averaged over a total of 10000 different random data streams per burst location for each burst length. The burst length is chosen once per simulation. Only burst lengths one through to seven are included as these were found to span the entire range of possible outcomes, increasing the burst length beyond this point does not yield a different result.

Writing 1s: In this implementation the error patterns are introduced through the writing 1s bit corruption technique which means the error patterns arriving at the receiver are affected by the data in the original message. The procedure follows a similar procedure as the inverting bits corruption technique in which the error array $e_i(x)$ is marked with consecutive 1s (depending on burst size) in the desired location, this is then interleaved resulting in the error message $e_t(x)$ with errors in the appropriate locations where they are to be introduced into the transmitted message. The range of burst lengths tested for this implementation is far greater than the inverting bits corruption technique, as the transition between the decoder being able to correct all errors, and being unable to correct any errors is much more gradual. Due to this the range of burst lengths tested is from burst length one to burst length 40 and averaged over 10000 random data streams per burst location for each burst length.

Writing 0s: In this implementation the errors are introduced using the writing 0s bit corruption technique and thus cannot ensure a consistent error pattern is received. To write 0s on the channel, the error array $e_i(x)$ is filled with ones, and then marked with consecutive 0s depending on the location of the burst. To ensure the burst is reconstructed by the deinterleaving process it is important that it is first interleaved with the GSM CCCH interleaving procedure before being introduced onto the channel. The simulation is run from a burst length of one to a burst length of 40, averaging the results over 10000 iterations per burst location for each burst length.

3.2.5 Proposed Signal Jammer Implementation (Single Decoding Error).

For this test the goal is to find the error pattern $e_c(x)$ which produces the least number of bits after the convolutional encoding process. Due to the GSM Fire decoder being able to detect any single burst errors with a length less than 40 bits, any burst error shorter than this remaining after the convolutional decoding process will be detected, and thus the packet discarded. The convolutional encoding is the last process that adds redundancy before transmission and as such the number of bits in the convolutionally encoded error message $e_i(x)$ is equal to the number of jamming bits required.

Inverting Bits: For this methodology to take advantage of the linear property of the convolutional code, allowing the error to propagate through the convolutional decoder undetected at the receiver, it is required that the encoded error message $e_i(x)$ is inter-leaved using the simulated GSM interleaver resulting in $e_t(x)$ and then introduced onto the channel by means of the XOR procedure. The other two bit corruption techniques cannot provide a consistent error pattern at the receiver and as such are not explored for this methodology. To create the desired error message, the error array $e_c(x)$ is first filled with 0s, the convolutional encoder has a memory length of five, and as such, any errors spaced more than five bits apart (errors with more than five consecutive bit positions between them) will not overlap and will always result in an increase in convolutionally encoded bits. Taking this into account this test is done by testing all possible combinations of five error bits in $e_c(x)$, starting from '00000' up until '11111'. This provides 32 (2^5) possible error combinations. These are introduced at all possible jamming locations starting from bit one up until 220. Each error pattern is tested at each location in the transmitted message. Due to the Fire decoder error detection procedure discarding a packet if errors are detected, this strategy aims at minimising the number of jamming bits required while ensuring an error is detected and the packet is discarded. The results are averaged over the 220 different error locations for each error pattern.

3.2.6 Proposed Signal Jammer Implementation (Generator Polynomial)

For this test the generator polynomial of the Fire code is used as the error pattern:

$$g(x) = (x^{23} + 1)(x^{17} + x^3 + 1) = (x^{40} + x^{26} + x^{23} + x^{17} + x^3 + 1). \tag{19}$$

Before convolutionally encoding the error pattern it is first shifted right by 40 bits, this is due to the Fire encoding being systematic, and so this shift moves the errors out of the parity bits and into the message bits (see *Figure 3.3*). Once the error pattern is shifted, it is then padded with 0s resulting in the 228 bit error message $e_c(x)$ as required by the convolutional encoder. The 228 bit error pattern is then fed into the convolutional encoder, resulting in a 456 bit encoded error pattern $e_i(x)$. This error pattern when convolutionally decoded results in the generator polynomial and thus should not be detected by the Fire decoder as a decoding error. This error message is then interleaved and XOR'd with the original encoded message $c_t(x)$, resulting in a 456 bit message with errors. It is notable that due to the nature of the convolutional encoder and decoder, the error pattern is message independent, and the convolutionally encoded 228 bit generator polynomial results in the same 456 bit error pattern $e_t(x)$ each time. This is what allows us to separate the jamming system from the GSM system until the last step before transmission $c_t(x)$ by replicating the GSM encoding and interleaving systems.

The process is repeated to test the effects of introducing multiple successive instances of this error pattern when creating $e_c(x)$. This is done by initially introducing just one instance as described above, then on each iteration adding an additional generator polynomial shifted right one bit from the previous iteration.

This method is the most custom tailored of the jamming techniques as it takes into account the underlying control channel packet structure being jammed. In *Section 2.1.3* it is discussed that for a mobile to make or receive calls it makes a channel request to the network and then waits for an Immediate Assignment message in response, assigning the mobile a dedicated channel. The Immediate Assignment message is identified by the mobile through a Request Reference parameter in the Immediate Assignment message. The goal of this jamming methodology is again to minimise the number of jamming bits required, although it provides the added benefit of remaining undetectable by the Fire decoder and thus not being discarded by the receiver. This is done whilst still corrupting the right bits to ensure communications cannot be made or received. To achieve this the generator polynomial is again used to hide the presence of errors from the receiver.

3.3 Conclusion

In this chapter the methodology used to complete this research and answer the research question is provided. The overall system setup is seen in *Figure 3.1* which is followed by a short discussion on the implementation of each jamming methodology to be compared. Chapter 4 presents the results of each of these tested methodologies separately, followed by a comparison between all three systems, and a comparison with recent relevant research.

Chapter 4: Results from Individual Jamming Methodologies

The following chapter describes the results from the jamming methodologies discussed in Chapter 3. Each of the jamming methodologies is individually analysed, and then compared according to the number of jamming bits each requires to cause communication failure on the channel.

Communication failure, as defined in *Section 1.2,* is the inability for any further calls or SMS's to be made on the channel. In this research this is achieved in two ways, the first is by ensuring that an error is detected at the receiver and thus the entire message is discarded. The second is by corrupting connection critical information within the message required by the MS for connection setup.

4.1 Random Error Locations

This experiment is implemented to test the limits of the CCCH forward error correction scheme against random errors on the channel. To consider this jamming methodology a success, 100% of the messages received should be detected as corrupt and as such discarded by the Fire decoder, i.e. no message should be received and correctly decoded at the receiver. To present this the number of jamming bits introduced (during transmission) vs the message error rate (MER) is graphed for each simulation. The MER is defined in this research as the number of incorrect messages (after decoding) received, divided by the total number of messages sent.

In this experiment, a varying number of errors are introduced at random locations in the transmitted message $c_t(x)$. The number of errors introduced is increased by one on each iteration up until the full message size of 456 bits is reached. The results for each error count is averaged over 10000 iterations to ensure consistent results. The results are analysed and discussed below.

4.1.1 Inverting Bits

Each 1 placed in the error message $e_t(x)$ ensures a bit is corrupted due to the XOR operation. The MER vs the number of bits inverted can be seen in *Figure 4.1.*

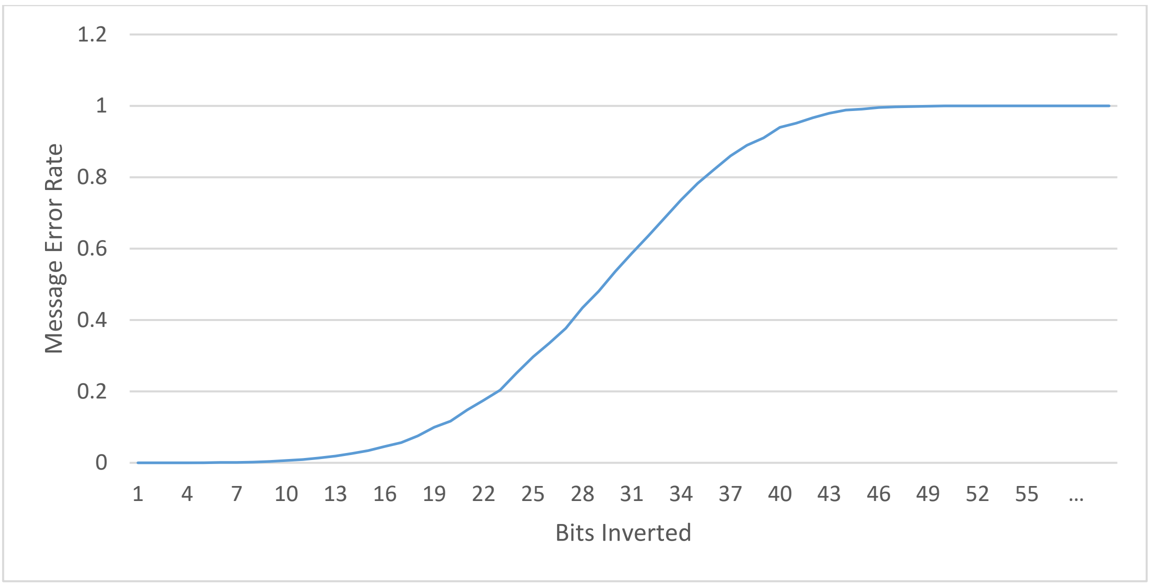

Figure 4.1: Message error rate vs bits inverted.

When looking at *Figure 4.1* it can be seen that the MER remains at 0 for the range of errors introduced from one up until five (5/456 = 1.10% of the transmitted message corrupted), this is where the initial limits of the code's error correction capabilities are reached. It then begins on a gradual incline during which only some of the received messages can be corrected, whereas the others are discarded by the receiver. This continues until the MER reaches 1 at the point where the number of bits corrupted is at 53 (53/456 = 11.62% of the transmitted message corrupted). Due to the random nature of this jamming approach, not all messages are successfully jammed, successful jamming is considered as when a MER of 1 is reached and remains consistent at this point for the rest of the simulation. To achieve this, 53 bits are required to be corrupt in the jamming channel, irrespective of their locations to result in 100% MER and effectively prevent communications on the channel. It can be noted for *Figure 4.1* that the full simulation continues until the point where the full 456 bits are introduced on the channel, this is not shown here however as the MER remains stable at 1 for the remainder of the bits inverted. This methodology achieves successful jamming requiring a total of 53 jamming bits to be introduced on the channel.

4.1.2 Writing 1s

Due to the transmitted message being of a binary nature, there is a chance that a 1 is introduced at a location where a 1 already exists, and thus the number of errors at the receiver is not always equal to the number of 1s introduced. The MER vs the number of 1s introduced can be seen in *Figure 4.2*.

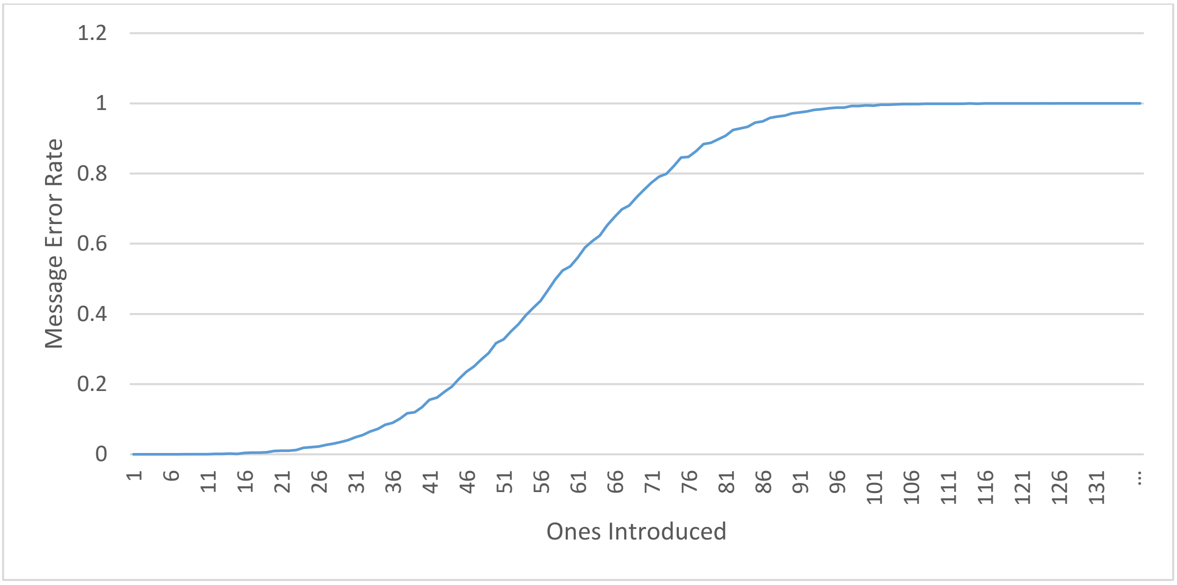

Figure 4.2: Message error rate vs 1s introduced.

Unlike in the previous test, the number of bits corrupted in the transmitted message $c_t(x)$ does not increase by one on each iteration when the number of errors written by the error message $e_t(x)$ is increased by one. This is due to some of the 1s being introduced where there are already 1s present. Due to the data being a random binary stream, this occurs on average for 50% of the bits written. In this simulation it can be seen that the initial number of bits written before message corruption is higher than that of the bit inversion technique simulated above, leaving a MER of 0 until the number of 1s written is equal to seven. From this point, as the number of errors introduced increases so does the MER, until it plateaus at a MER of 1 at the point where 124 1s are written and an average of 61.92 bits have been corrupt ($62/456 = 13.60\%$ of the transmitted message corrupted). It is again notable that this simulation was run until the full 456 1s were introduced, but the MER remained at 1 for the rest of the simulation and as such is not shown. Successful jamming is achieved at the point where 124 1s are written, resulting in 62 bits being corrupt on the air. This is more than double the number of jamming bits required by the inverting bits method.

4.1.3 Writing 0s

Introducing 0s into binary data means that the position being written with a 0 may already be a 0 and thus no change is made, again this has on average a 50% probability for random binary data. While running this simulation an interesting trend in the number of errors reported by the Fire code's error detection mechanism was noticed. To show this the probability of an error being detected by the Fire decoder vs the number of 0s written is shown in *Figure 4.3*.

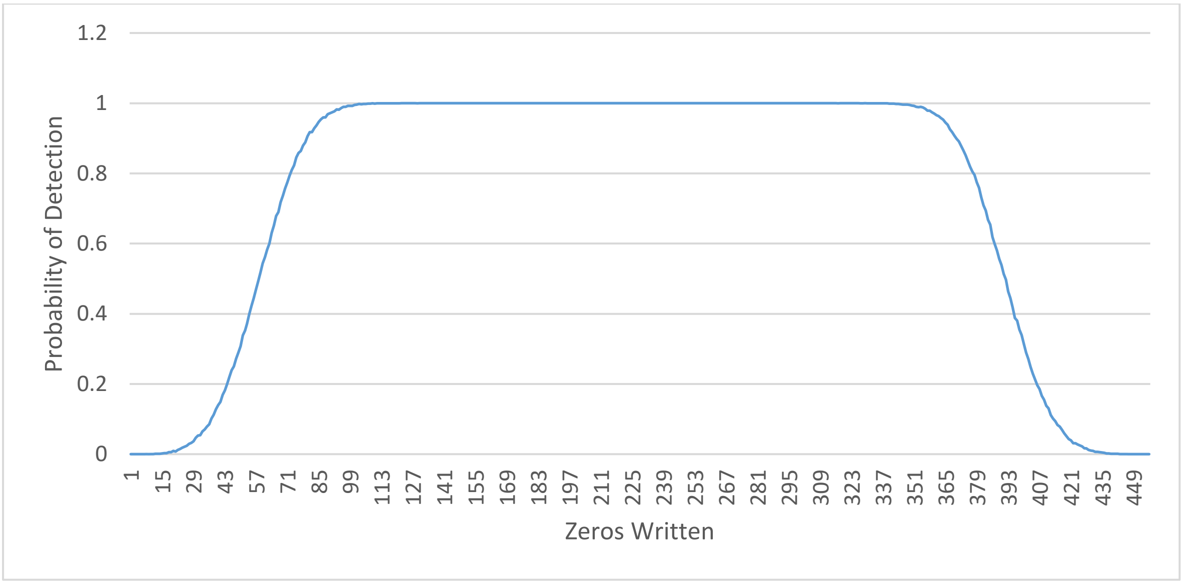

Figure 4.3: Probability of detection vs 0s written.

This simulation produced interesting results, once the probability of detection reached its maximum of 1, it did not remain like this for the rest of the experiment as expected, instead it dropped back down to zero before the end of the simulation. This is due to the fact that the Fire decoder error detection method does not detect that there is an error in the message if both the parity and message parts of the received codeword are all 0s (as shown in *Section 2.2.1*). This is an interesting result, but would require for the entire transmitted message to be overwritten with 0s, which requires all 456 bits to be corrupted, and as such is not a viable solution.

The MER vs the number of 0s written is shown in *Figure 4.4*.

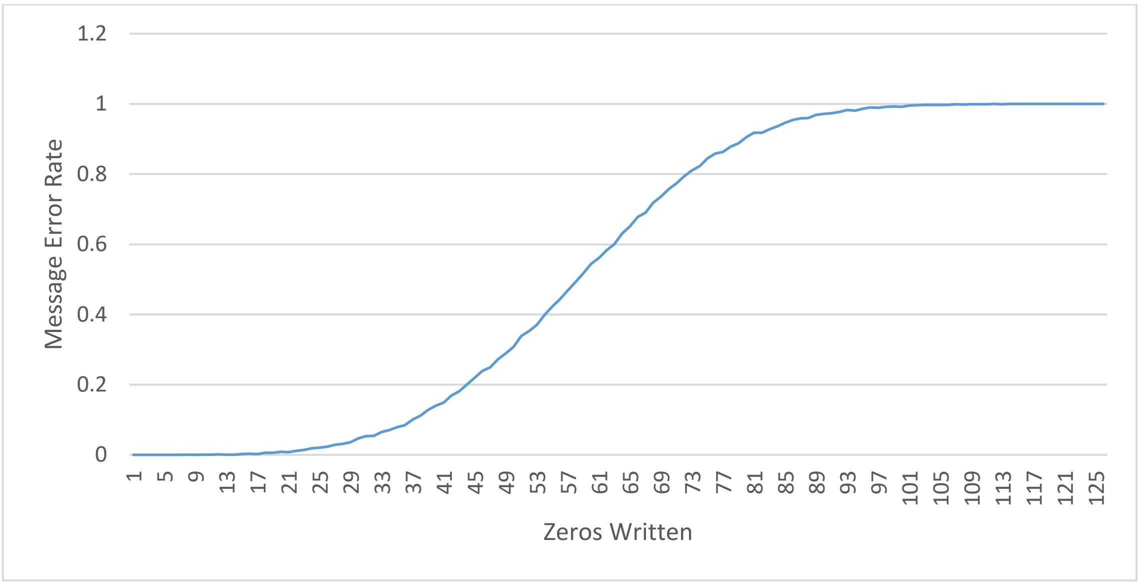

Figure 4.4: Message erasure rate vs 0s written.

It can be seen that a similar trend is followed as that for the previous simulations in this category in which the MER remains at 0 initially until the number of 0s written reaches a specific value, in this case eight, and then continues on a smooth (but not linear) incline until it reaches a MER of 1 when the number of 0s written is 122, at this point the number of errors introduced is 60.94 (61/456 = 13.38% of the transmitted message corrupted). Successful jamming in this writing zeroes simulation is achieved at the point where 122 0s are written, resulting in 61 bits on average being corrupt on the air.

Each of the error corruption techniques is tested and it is found that the bit inversion technique requires the least number of bits (53) to be written over the original message, requiring less than half of that of the writing 0s (122 bits written) and writing 1s (124 bits written) approaches.

4.2 Errors in Bursts

The simulation is run for varying burst lengths ranging from a minimum of one up until a maximum of 40 (depending on bit corruption technique in use). For each iteration the errors are introduced into 10000 random binary streams and the results averaged, so as to provide accurate and consistent results. For each burst length the positioning of the error burst is shifted across the entire transmitted message, starting at the first bit in the random data stream and progressing through till the last bit it can without going past the end of the message.

4.3 Inverting Bits

This error corruption technique ensures an error occurs for each bit introduced. When introducing errors in bursts, this approach can ensure that every bit in the burst is corrupted. The two noteworthy burst error lengths achieving the best results in this simulation are burst length four and burst length five. The MER vs the burst error location (the location of the first bit in the burst error) for error lengths four and five are shown in *Figure 4.5* and *Figure 4.6* respectively.

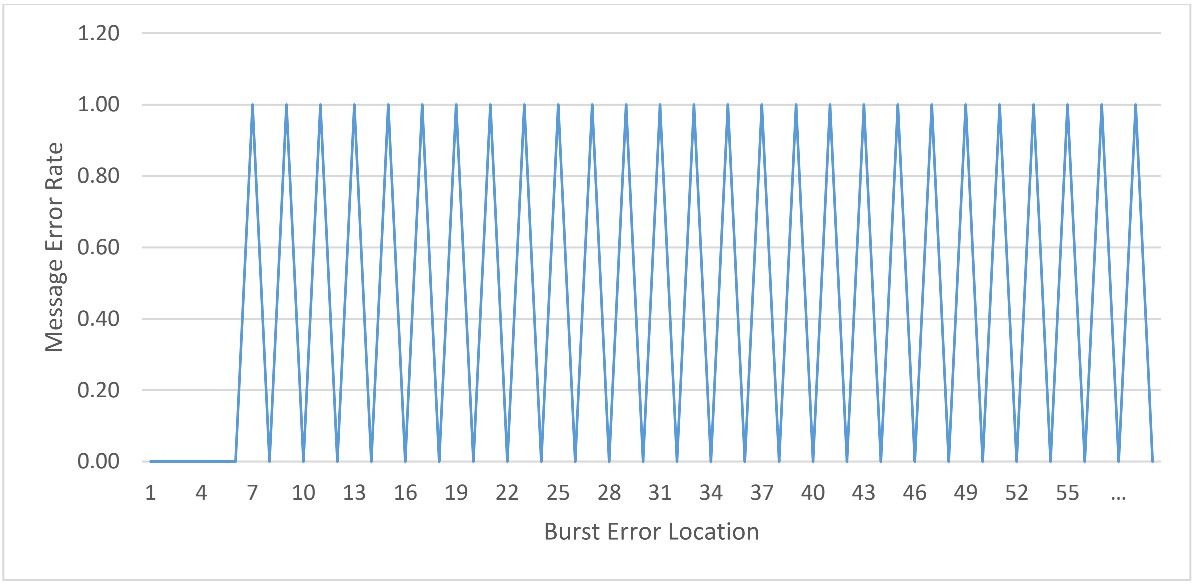

Figure 4.5: Message error rate vs burst error location for error length 4.

The reason for the choice of burst error lengths four and five is that combined they cover the entire transition between the decoder being able to correctly decode all of the received messages, and the decoder being unable to correctly decode any of the received messages. In *Figure 4.5* the results for burst error length four are shown, this is the first point in which the decoder starts being unable to correctly decode all received messages. An interesting result occurs from this burst error length due to the convolutional encoder using two different generator polynomials to encode the data on even and odd bit numbers. This results in even and odd bit numbers each being encoded and decoded with a different bit combination (see *Figure 2.3).* Depending on the bit combination of each encoded bit a different outcome from the code's error correcting capability is achieved depending on whether the burst error occurs starting on an even or an odd bit number. When looking at Figure 4.5 the following can be said: When the burst errors first bit is positioned on an even bit number the decoder is able to correct all the received messages without errors irrespective of the message being sent. When the burst error occurs on an odd bit number the decoder is unable to decode any of the received messages again without influence from the data in the message being sent. An exception to this occurs for burst error locations one through to five where the decoder is always able to correctly decode all received messages. Therefore any error pattern which results in a burst error of length four after the deinterleaving process is capable of effectively jamming the channel as long as the resulting burst is positioned on any odd bit number greater than five. Therefore this method requires a total of four bits to be transmitted to cease communication on the channel.

The next burst error length discussed is burst error length five. In *Figure 4.6* the MER vs the burst error location is graphed for error length five.

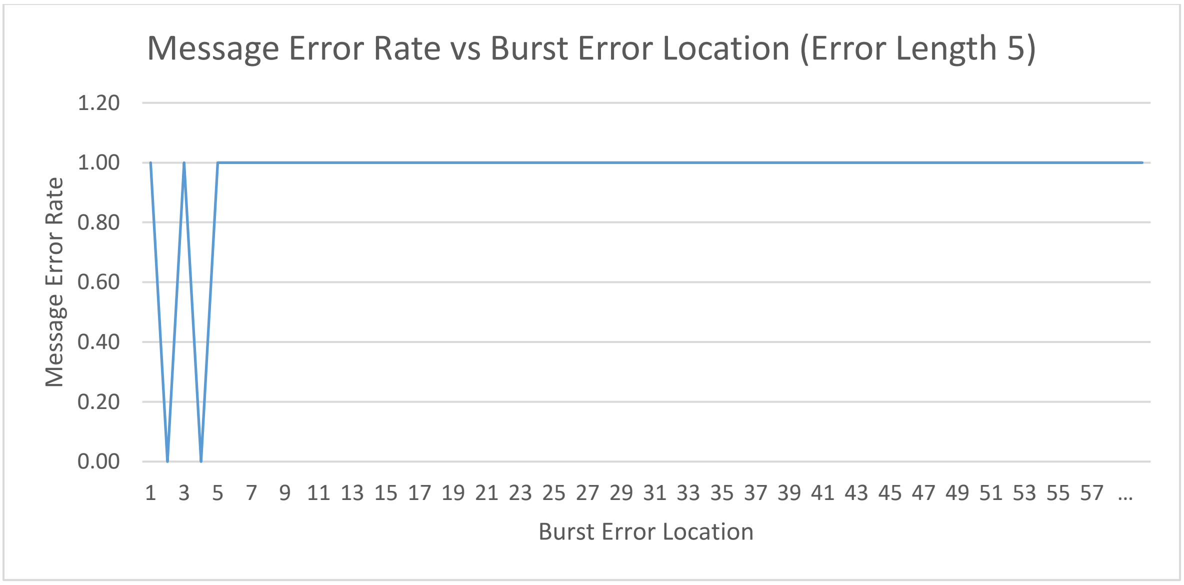

Figure 4.6: Message error rate vs burst error location for error length 5.

When analysing *Figure 4.6*, its apparent that it follows a similar trend to that of *Figure 4.5* in which the MER displays an alternating behaviour up until the burst error location reaches bit five, then a consistent behaviour till the end. For burst error length five the trend in the MER for starting positions one up until five are consistent with the results found in the previous simulation for burst error length four in which burst errors starting on the even bit numbers are decoded correctly whilst burst errors starting on odd bit numbers are decoded incorrectly. In this simulation the MER alternated between zero and one for even and odd burst starting locations from one up until five, then remained stable at one for the remainder of the error locations from bit six until bit 456. For the burst error of length five, successful jamming is achieved by ensuring the starting bit position of the error burst is greater than five, or lies on any of the even bit numbers in the received message after the deinterleaving process. When comparing this to a burst length of four, the longer error burst provides more possible jamming patterns, while the shorter error burst requires less jamming bits at the cost of having a smaller variety of possible jamming patterns.

By using a burst error of length four its possible to effectively ensure that no messages are decoded correctly at the receiver if the four jamming bits are correctly positioned so that the first bit of the resulting error burst lies on an odd bit position greater than bit five in the transmitted message.

4.3.1 Writing 1s

This error corruption technique no longer ensures that an error occurs at each location where a jamming bit is transmitted. Due to this technique writing 1s irrespective of the original bit being transmitted, there is no guarantee that a 1 won't be written where there is already a 1 present in the message. This means that even though the jamming patterns transmitted are still full bursts the resulting error pattern in the received message cannot be guaranteed. The experiment is run for increasing burst error lengths until an MER of 1 is consistently achieved at the receiver.

Due to the inconsistent nature of the effects of this bit corruption technique, the number of jamming bits required is much higher than that of the bit inversion technique with a much slower transition between the initial MER of 0 and 1 resulting in almost constant MER per burst length. To effectively show this transition the results are averaged for each burst length and the MER vs the burst length for the writing 1s bit corruption technique is shown in *Figure 4.7.*

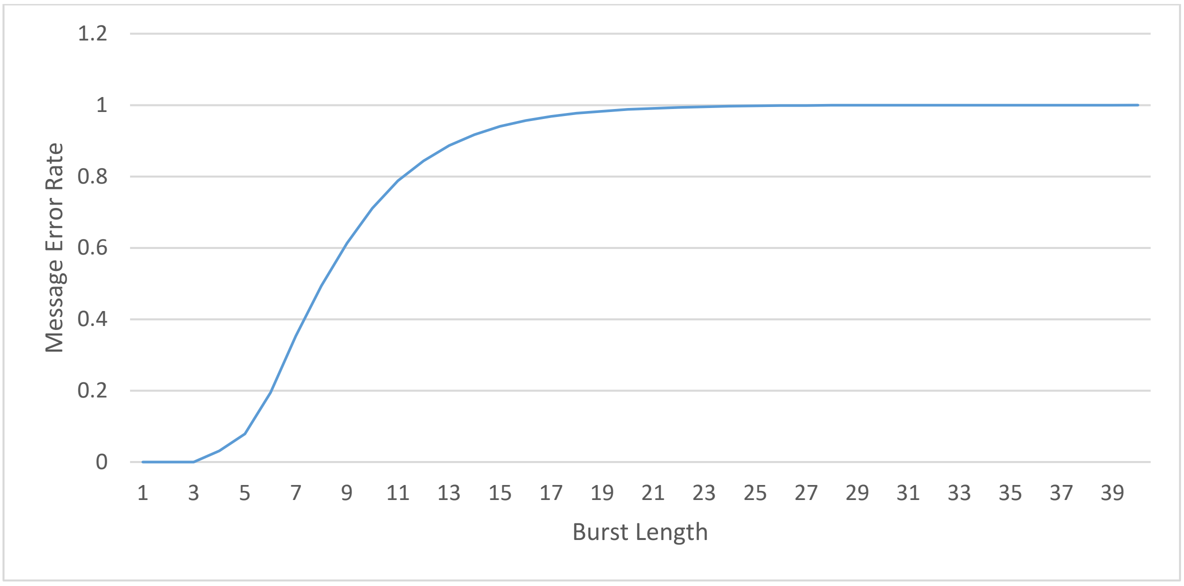

Figure 4.7: Message error rate vs burst length (1s).

When comparing these results to the ones obtained when using bit inversion as the bit corruption technique, it is immediately obvious that the results are highly influenced by the data being transmitted on the channel. This can be seen in *Figure 4.7* where the slope of the MER increases in a nonlinear fashion, as the number of bits being transmitted (burst length) is increased, the number of messages arriving correctly at the receiver decreases. This slope has a steep incline, and then levels off tending very slowly towards an MER of 1, eventually reaching a consistent MER of 1 at a burst length of 37. Even though the burst length required before a consistent MER is reached is much higher than that of the bit inversion corruption technique, it is notable that the MER has already reached 0.99 at a burst length of 21. Therefore successful jamming is achieved at a burst length of 37 bits.

4.3.2 Writing 0s

As with the writing 1s bit corruption technique for burst errors, the writing 0s bit corruption technique is again unable to ensure a consistent error pattern is achieved in the received message. As a result of this, a similar trend occurs where the transition between an MER of 0 and 1 happens gradually over many error burst lengths, without any significant variation per burst. To account for this the results are averaged per burst length and the average MER vs the burst length is shown in *Figure 4.8*.

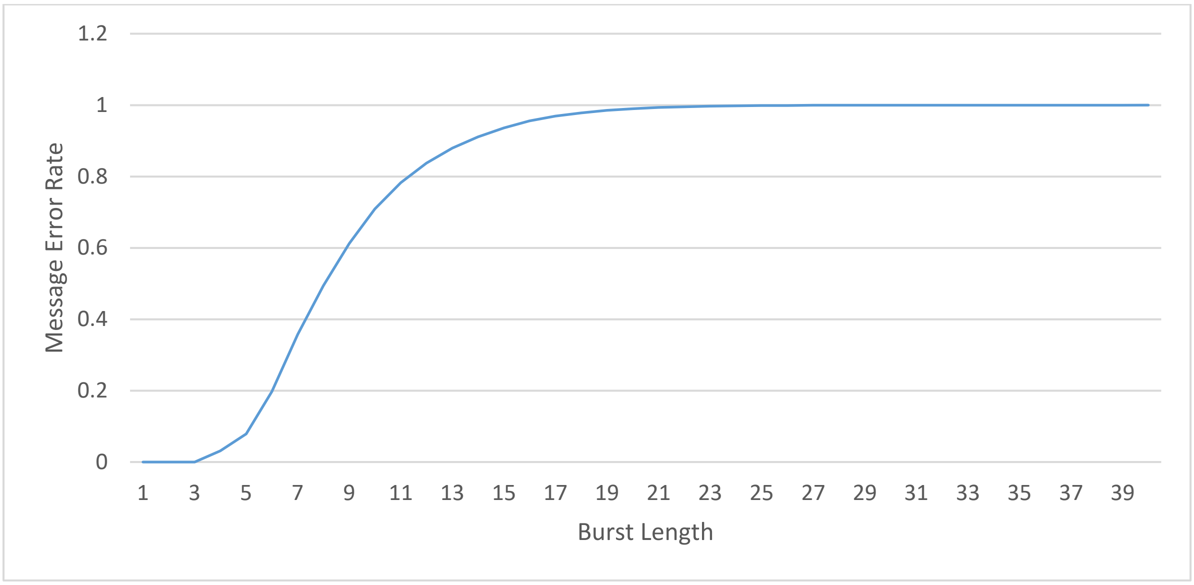

Figure 4.8: Message error rate vs burst length (0s)

When analysing the results shown in *Figure 4.8* it is seen that the curve is almost identical to that shown in *Figure 4.7* where the bits being written are 1s. The MER for this simulation again reached an MER of 0.99 at a burst length of 21 bits, and reached a stable MER of 1 at a burst length of 37 bits. Therefore this jamming methodology requires a total of 37 jamming bits to be introduced to cause communication failure on the channel.

4.4 Proposed Signal Jammer Implementations

The two proposed signal jammer designs below both take advantage of the linear property of the convolutional code which is used in the GSM CCCH system. This property allows for error patterns to be created capable of bypassing the Viterbi decoder without the errors being detected or corrected by the decoder (discussed in *Section 3.1.4*). The advantage to this is that error patterns can be created which target weaknesses in the Fire code error detection procedure, as this is where the errors will first be detected. Due to the strict requirements of these error patterns arriving in the correct form at the receiver, only the bit inversion bit corruption technique is applicable for these simulations as it can ensure a consistent error pattern is achieved in the received message.

4.4.1 Single Decoding Error

The single decoding error proposed jamming approach is one which takes advantage of the way in which the GSM system handles errors in the received message. The name of this experiment was chosen as such as the initial goal was to try reducing the required jamming bits to only one bit. The first step in error control is the Viterbi decoder which checks for and attempts to correct errors in the received message, the second step is the Fire decoder which then checks for any remaining errors present in the received message, if any remaining error is found, instead of attempting to correct the errors like the Viterbi decoder does, the message is discarded by the Fire decoder which then returns a decoding failure. This approach aims to minimise the number of jamming bits required by creating error patterns which pass through the Viterbi decoder undetected, but are then detected at the next step by the Fire decoder causing the message to be discarded. By doing this and using the bit inversion corruption technique, the jamming system is no longer required to beat the error correction capabilities of the GSM convolutional code, but is instead only required to use error patterns which are valid codewords of the convolutional code, so that these errors traverse through the Viterbi decoder undetected but cause decoding failure at the Fire decoder. To ensure the error patterns introduced are valid codewords after interleaving, the error patterns are passed through the replica GSM convolutional encoder and interleaver and the resulting bits are then introduced onto the channel. The resulting number of bits after convolutional encoding is equal to the number of jamming bits required. To test all possible combinations of five bits (the reason for the choice of five bits is discussed in *Section 3.2.5*), the error pattern for each iteration is set equal to the binary representation of that iteration number with the most significant bit first, starting at sequence number zero and progressing through to sequence number 31. For example: error sequence number five has the error pattern '00101'. The number of jamming bits required (after encoding the error sequence) vs the error sequence number is shown in *Figure 4.9.*

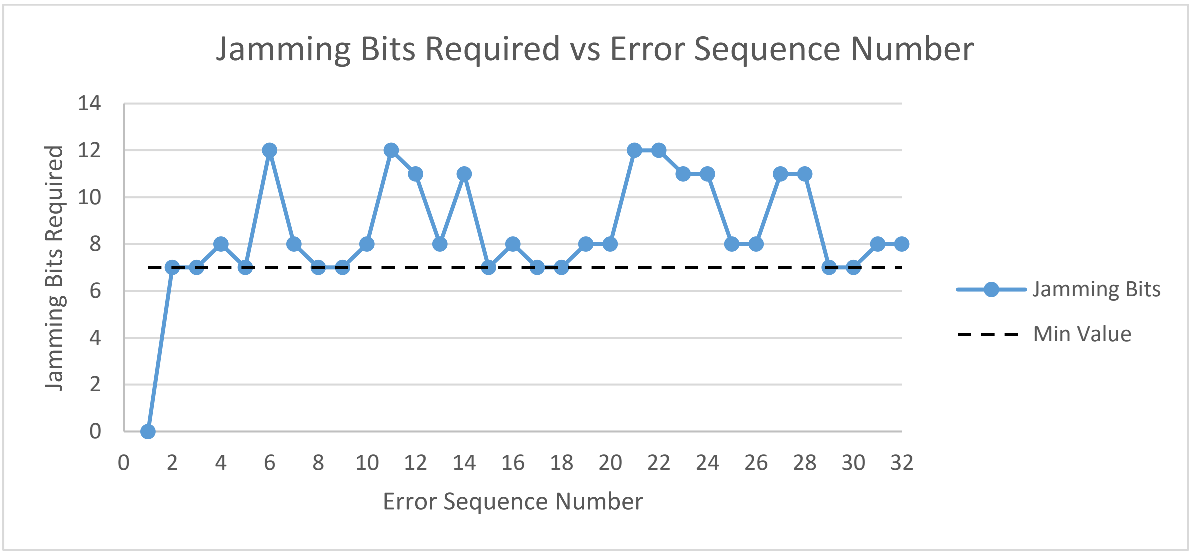

Figure 4.9: Jamming bits required vs error sequence number.

Due to the error pattern bypassing the convolutional decoder undetected and then being detected by the Fire decoder, the location of the error is no longer a concern and the error is detected irrespective of its location. To test this the errors were shifted through all possible locations and the results averaged. A line is drawn through the error sequence numbers which minimize the number of jamming bits required. The minimum number of jamming bits required to ensure at least one error is present at the Fire decoder is seven bits. There are multiple error sequence numbers on which this occurs, and any one of which can be chosen to produce this result, irrespective of the location of the error in the message.

Therefore to effectively cease communication on the channel, this jamming approach requires a minimum of seven jamming bits to be transmitted on the channel.

4.4.2 Generator Polynomial

In all the jamming strategies above the jammer relies on the Fire decoder at the receiver detecting an error and then discarding the message. In this jamming strategy we propose an alternative method of exploiting the GSM Fire decoder, one which again exploits the linear property of this code. When looking at the above example we can see that it is possible to hide the presence of errors from the Viterbi decoder by using a valid codeword from the GSM convolutional code as an error pattern. In this approach we take the same idea and extend it further to include the Fire decoder, by providing an error pattern which is both a valid convolutional codeword, and when decoded results in a valid Fire codeword (in this case the generator polynomial used in the construction of the Fire code), the presence of these errors can be hidden completely from the receiver. If the presence of these errors is hidden from the receiver however, the received message will not be discarded by the Fire decoder and as such it is of crucial importance that the errors are introduced in locations which contain information critical to the user's ability to successfully communicate on the channel.

This experiment tests the effects of using the generator polynomial in the creation of the error pattern and what effect introducing multiple of these error patterns would have on the original message. The simulation starts with only one generator polynomial written in $e_c(x)$ aligned with its first bit to be written at bit location 41, and then on each iteration writes an additional generator polynomial at the next sequential bit location, so in the second iteration two generator polynomials will be written aligned with their first bits at location 41 and 42 in $e_c(x)$. The initial shift to bit location 41 is to move the errors out of the parity section and into the data section of the received message (see *Figure 3.3*). On each iteration the number of sequential generator polynomials written is equal to the iteration number. An alternate method of introducing the polynomials is also tested in which one generator polynomial has a fixed location and the other is shifted through the message, combining them in this fashion always results in an increased number of required jamming bits and as such the results are not discussed. The error patterns are created before the convolutional encoding and interleaving stage, and as such the error locations refer to the location of the first bit of the generator polynomial written in $e_c(x)$. In *Figure 4.10* the last error location on the horizontal axis refers to the location of the last generator polynomial written, e.g. a last error location of 43 means that generator polynomials are written starting at bit numbers 41,42 and 43 in $e_c(x)$.

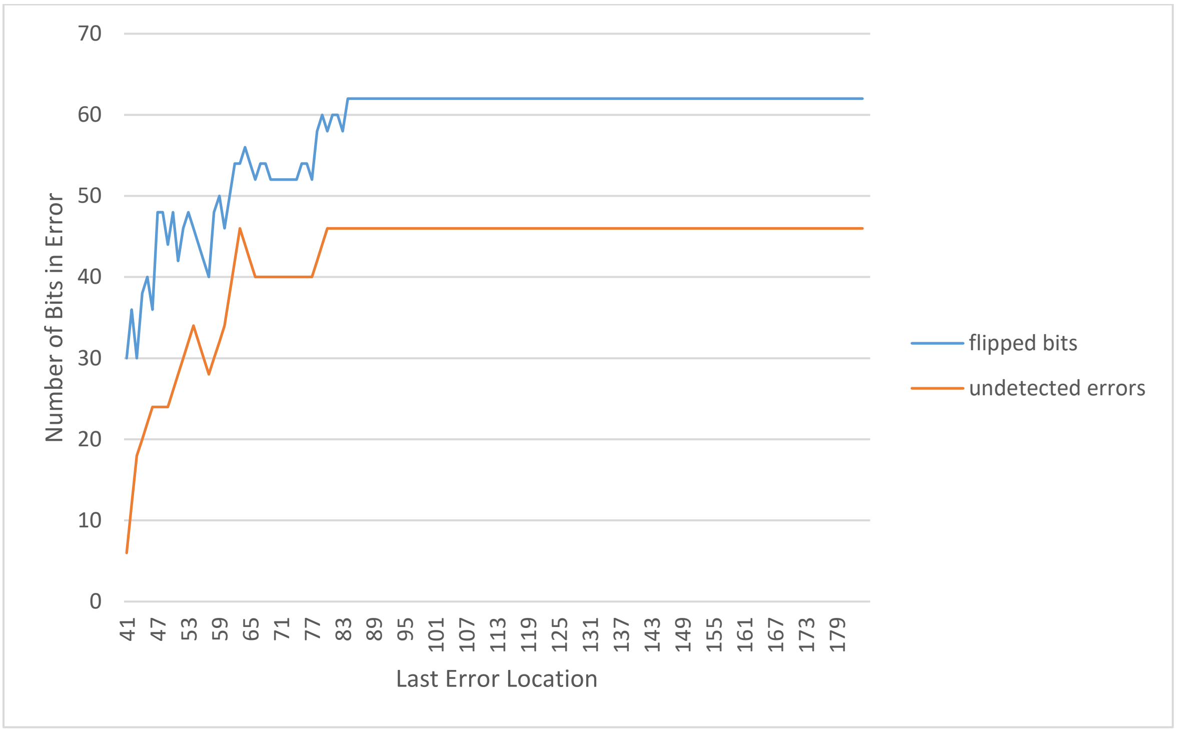

Figure 4.10: Number of bits in error vs last error location.

In *Figure 4.10* it is seen that by introducing multiple generator polynomial error patterns the number of undetected errors introduced can be increased, however this comes at an increased cost, as the number of jamming bits required generally increases with it. An exception to this occurs however at the point where three consecutive generator polynomials are used as the error pattern, at this point the number of jamming bits required returns back to its initial value of 30, but the number of undetected errors introduced does the opposite and increases from its initial value of six bits to 18 bits. This means that triple the amount of bit errors can be introduced without requiring an increased number of jamming bits to be transmitted. This is higher than the previous results but provides the advantage of not being detected by the receivers forward error correction.

Therefore this jamming strategy requires a minimum of 30 bits to be written to ensure the errors introduced are not detected by the receiver, depending on the combination of these 30 bits they result in a total of either six or 18 corrupt bits in the received message. The receiver is unable to detect these errors and will not discard the received message, it is due to this that the location of the introduced errors is of crucial importance to the success of this jamming strategy.

To this end the following strategy is explored: From the results above it can be seen that it is possible to insert errors in such a way that the receiver is unable to detect the presence of these errors. It can also be seen that the minimum number of jamming bits required to avoid detection is 30, which can result in either six or 18 undetectable errors after the decoding process. These six errors are not sequential but are spread over a total of 41 bits (43 bits for 18 errors), they are however consistent and thus can be placed at specific locations depending on the required error locations. As discussed in *Section 2.1.3* the Request Reference Information Element is a 3 byte long field (occupying bit numbers 80-103) that is responsible for identifying the Immediate Assignment message to the mobile subscriber, without this the mobile device has no way of identifying that it is the intended recipient of the Immediate Assignment message, and thus does not know to continue with the connection setup. The field before it which is also 3 bytes long occupies bit numbers 56-79 and contains the Packet Channel Description. The Packet Channel Description is also required for successful connection setup, and contains information such as: on which channel the connection setup will continue, the type of channel, and whether frequency hopping is in use. Due to the errors created by the generator polynomial being spread over a span of 41 bits it is chosen that the errors are spread between the Request Reference and Packet Channel Description Information Elements. The possible error positions for the six and 18 error jamming approaches are shown in *Figure 4.11* and *Figure 4.12*, with a summary of the damage done in each position shown in *Table 4.1* and *Table 4.2* respectively.

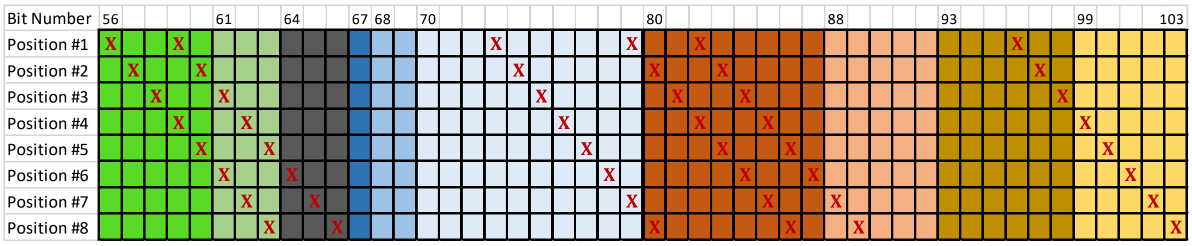

Figure 4.11: Positioning of 6 error bits in Paging message.

When using only a single generator polynomial for the error, there are eight possible locations it can be placed between bit 56 and bit 103. Each error position results in different parts of the message being corrupt to different extents, the breakdown of which is shown in *Table 4.1.*

Table 4.1: The number of corrupt bits in each Immediate Assignment information element per error location (6 errors)

		Corrupt Bits per Error Location (bits)							
Key	**Description**	**#1**	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**
🟩	Channel Type	2	2	1	1	1	0	0	0
🟩	Timeslot	0	0	1	1	1	1	1	1
⬛	Training Sequence	0	0	0	0	0	1	1	1
🟦	Hopping Channel	0	0	0	0	0	0	0	0
🟦	Spare/MAIO	0	0	0	0	0	0	0	0
🟦	ARFCN's/HSN	2	1	1	1	1	1	1	1
🟧	Random Access Information	1	2	2	2	2	2	1	1
🟧	T1'	0	0	0	0	0	0	1	1
🟨	T3	1	1	1	0	0	0	0	0
🟨	T2	0	0	0	1	1	1	1	1

When analysing the above jamming pattern locations the goal is to ensure that errors are introduced in locations ensuring critical connection setup information is corrupt by the jammer. Due to the number of required jamming bits being a constant, only the locations of the resulting error positions can be varied, with all of the possible positions above corrupting bits in the Random Access Information any one will result in successful jamming. As this approach is directed towards minimising required jamming bits with reduced detectability the location of the jamming pattern is chosen in such a way as to alter values which are least likely to leave their acceptable ranges (incorrect but valid), thus limiting the possible number of invalid values detected by the receiver. Jamming strategy #6 is chosen as the best location to achieve this goal as it maximises damage in Random Access Information, and doesn't affect channel type.

The results when using three consecutive generator polynomials as an error pattern are shown in *Figure 4.12*, and the number of corrupt bits per information element is shown in *Table 4.2*.

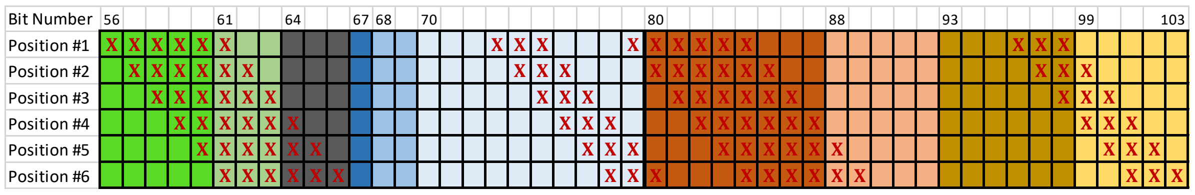

Figure 4.12: Positioning of 18 error bits in Paging Message

Table 4.2: The number of corrupt bits in each Immediate Assignment information element per error location (18 errors)

		Corrupt Bits per Error Location (bits)					
Key	**Description**	**#1**	**#2**	**#3**	**#4**	**#5**	**#6**
🟩	Channel Type	5	4	3	2	1	0
🟩	Timeslot	1	2	3	3	3	3
⬛	Training Sequence	0	0	0	1	2	3
🟦	Hopping Channel	0	0	0	0	0	0
🟦	Spare/MAIO	0	0	0	0	0	0
🟦	ARFCN's/HSN	4	3	3	3	3	2
🟧	Random Access Information	5	6	6	6	5	5
🟧	T1'	0	0	0	0	1	2
🟫	T3	3	2	1	0	0	0
🟨	T2	0	1	2	3	3	3

For the simulation with 18 corrupt bits, each of the different possible error positions still ensures the Random Access Information is corrupt, and as such the choice of any of the above locations results in successful jamming. Therefore this jamming system can be designed for either introducing six or 18 undetectable errors after decoding, and requires 30 jamming bits to be transmitted on the channel.

This jamming approach is capable of effectively ceasing communication on the channel requiring a total of 30 jamming bits to be transmitted.

4.5 Evaluation and Comparison of All Jamming Methodologies

Up to this point this chapter has explored each of the jamming methodologies simulation results individually and provided a discussion on each methodology's performance such as where each system reaches its effective jamming capability. In each of the tests the goal is to minimize the number of jamming bits required, in this section the results from all jamming strategies are contrasted providing a side by side comparison of the minimum number of jamming bits required for each.

Note, the following analysis is made with respect to CCCH jamming only, the CCCH packets take up 36 of the total 51 packets ($36/51 = 70.59\%$) to be transmitted on timeslot 0 (see *Figure 2.1*), and thus common control channel jamming already offers an improvement of 29.41% over conventional always on methods (when considering only timeslot 0). Due to this investigation focussing on CCCH jamming, instead of using always on jamming as a baseline of comparison, the baseline used for comparison is an always on version of a CCCH jammer which we denote $CCCH_A$ jamming and define as a jammer which transmits for all 456 information bits in every group of CCCH bursts. The comparison below is done with respect to a 456 bit CCCH message.

The minimum number of jamming bits required is shown for each of the jamming methodologies and bit corruption techniques in *Table 4.3*.

Table 4.3: Minimum number of jamming bits required for all jamming methodologies and bit corruption techniques.

Jamming Methodology	**Bit Corruption Technique**	**Jamming Bits Required (bits)**	**Improvement over $CCCH_A$ jamming (%)**
Random Error Locations	Inverting Bits	53	88.38
	Writing Ones	124	72.81
	Writing Zeros	122	73.25
Errors in Bursts	Inverting Bits	4	99.12
	Writing Ones	37	91.89
	Writing Zeros	37	91.89
Single Decoding Error	Inverting Bits	7	98.46
Generator Polynomial	Inverting Bits	30	93.42
Minimum bits	Inverting Bits	4	99.12
Maximum bits	Inverting Bits	124	72.81

When comparing the above results it is immediately apparent that the use of different bit corruption techniques significantly impacts the number of jamming bits required to cease communication on the channel. It is also apparent that by taking the parameters of the GSM CCCH into consideration when designing a jamming scheme, the number of required jamming bits can be greatly reduced. The inverting bits corruption technique significantly outperforms the other two bit corruption techniques as it can ensure the location of the errors in the received messages are consistent. To do this the inverting bits corruption technique requires significantly more advanced and expensive hardware, whereas the two other bit corruption techniques can be implemented on basic and inexpensive equipment, as they do not have the same strict processing requirements. Therefore, for a fair comparison to be made between the jamming systems, the results are grouped and compared according to the bit corruption technique in use.

4.5.1 Inverting Bits

The inverting bits corruption technique is used in all four of the jamming methodologies tested as it is the only bit corruption technique capable of ensuring that each bit introduced results in an error at that location in the received message, and therefore being able to ensure a consistent error pattern at the receiver. The first methodology to be discussed is the random error locations methodology in which a predefined number of jamming bits are written at random locations in the transmitted message. This is the most basic jamming approach implemented where the only knowledge the jammer has of the system is the positions of the GSM CCCH packets. This methodology requires the highest number of jamming bits to cease communication, but still shows an improvement of 88% over an always on CCCH_A jammer. The second methodology to be discussed is the burst error jamming approach, which takes into account both the positions of the GSM CCCH packets as well as the interleaving procedure in use. By designing error patterns which are known to result in bursts after the deinterleaving process it allows for the number of jamming bits required to be reduced from the 53 bits required in the random approach, to only four (properly positioned) bits using the burst error approach (which is 7.55% of the jamming bits required in the random jamming approach). This shows that by increasing the knowledge of the protocol in use we can significantly reduce the number of jamming bits required to achieve successful jamming. The next methodology to be discussed is the proposed single decoding error approach, in which the system takes into account the error correction encoding procedures used on the channel. By designing the error pattern to bypass the convolutional decoder undetected targeting the Fire code, the minimum number of jamming bits required can be reduced to only seven bits (13.21% of the bits required by the random jamming approach). The last jamming methodology to be discussed is the proposed generator polynomial approach in which we attempt to minimise the number of jamming bits required to cease communication on the channel, while hiding the presence of the errors at the receiver. This is provided as an alternate solution to minimizing the number of jamming bits required as it reduces the chances of the jammer being detected by creating jamming patterns which do not cause constant decoding errors at the receiver. In this jamming approach we are able to effective cease communication requiring only 30 bits to be transmitted (56.6% of the bits required for random jamming approach).

Both the burst error and single decoding error tests produced significant reductions over CCCH_A jamming in the number of transmitted jamming bits required with the burst error approach requiring a minimum of four jamming bits to effectively cease communication on the channel. This is an improvement of 99.12% over CCCH_A jamming.

4.5.2 Writing 1s

The second bit corruption technique tested involves replacing the bits on the air with 1s irrespective of the original bit being transmitted. This is provided as a more viable economical solution as it can be achieved without the need for expensive hardware. When using this bit corruption technique to jam the channel the random error location jamming approach again requires the highest number of transmitted jamming bits requiring 124 1s to be written. The burst error jamming approach is able to improve on this providing effective jamming requiring only 37 jamming bits to be introduced. This is only 29% of the required bits for random jamming, and a 91.89% improvement over CCCH_A jamming. In a real life situation, depending on how strict the requirements of the jamming system are, this approach reached 99% MER requiring only 21 jamming bits to be introduced.

4.5.3 Writing 0s

The third bit corruption technique tested is a variation of the writing 1s bit corruption technique in which the errors introduced are introduced by replacing the original bit with a 0. The results are very similar to those achieved with the writing 1s approach with the two results converging as the number of iterations over which the results are average is increased. When averaging 10000 iterations the random error location jamming approach achieved successful jamming while writing 0s requiring 122 jamming bits to be introduced, this is two bits less than what is required for the writing 1s approach. For the burst error approach the error introduced were less random and as such the number of jamming bits required converged to a stable 37 bits for both the writing 0s and writing 1s bit corruption techniques.

4.5.4 Comparison of Jamming Strategies Against Previous Research Done.

In this research various different jamming strategies are discussed, this chapter covers all the tests that are done for this research on intelligent CCCH jamming strategies which take into account the forward error correction scheme in use, this section compares the results obtained above to research done prior to this point on intelligent signal jamming. The first paragraph outlines the results found in literature, whereas the second provides the comparison with this research.

The results achieved in this paper are compared to a similar paper by Petracca et al. in [18], in which three jamming procedures are tested also focussed on the attacking the GSM control channel. The work focusses on jamming the synchronisation control channels, let it be noted that the initial jamming time of 5.296s is left out intentionally, this is so a fair comparison can be made between the two systems, as this a representation of the continuous usage requirements. The three procedures investigated are FCCH jamming, SCH jamming, and BCCH jamming. Due to this paper focussing on SNR in the presence of AWGN noise, the required jamming times for comparison are calculated assuming a jammer which always transmits for the full duration of each of those channels. The FCCH jamming procedure blocked all communications requiring 2.89ms jamming time per 51 frame multi-frame (235.4ms), the same as that for the SCH jamming. BCCH jamming only required 2.31ms per 51 frame multi-frame. As BCCH jamming requires the shortest jamming time it will be used as the basis of comparison.

Using (1) the time required for each of the jamming methodologies tested in this paper can be calculated. Due to the testing being done on a bit level, this is only done so a comparison can be made. The minimized number of jamming bits required for the bit inversion technique is four bits per CCCH group, which requires a transmission time of 0.13ms (5.63% of the time required for BCCH jamming) per 51 frame multi-frame. In the writing 1s approach, the minimized number of jamming bits required is 37 bits (same as that for the writing 0s approach), which requires a transmission time of 1.23ms (53.25% of the time required for BCCH jamming) per 51 frame multi-frame.

This system does have limitations however and is unable to block calls that have already been connected, so as with in this paper it can be achieved with an additional initial constant jamming duration of 5.296s.

Chapter 5: Conclusion

5.1 Research Summary

In this research the following question is asked: "*How can we exploit the control channel forward error correction scheme of the GSM system to minimize the number of jamming bits required to prevent communications on the channel?*". In this dissertation the research undertaken to answering this question is discussed. An in depth literature review is done into the commonly used jamming methodologies, as well as what research has been done thus far to improve on these methodologies. This is followed by an investigation into the GSM protocol, providing insight into important operations occurring in the data link layer for control channels, including the time diversity and error control coding techniques. The research methodology followed is outlined in Chapter 3 which describes in detail each of the jamming methodologies which were tested in the research as well as the software used for the simulations. The results of these simulations are presented in Chapter 4, where the results from each jamming methodology is first individually analysed, and then compared and contrasted against each of the other jamming strategies and finally against previous work done in the same field of research.

5.2 Achievements

In attempt to minimise the number of jamming bits required to prevent communications by exploiting the GSM control channel forward error correction scheme, four jamming methodologies are presented. The first methodology presented in which errors are introduced at random locations in the message is the most trivial jamming approach, as it tests the control channel FEC scheme without the use of custom error patterns. Each proceeding methodology aims to expand on this by attempting to exploit different components of the forward error correction scheme, this is done by using custom error patterns. Three bit corruption techniques are also tested which involves 1) inverting of bits on the air, 2) the writing of 1s and 3) the writing of 0s, the bit inversion technique significantly outperforms the other two techniques as it can ensure consistent error patterns at the receiver. By exploiting different components of the control channel FEC scheme each of the jamming methodologies is capable of significantly reducing the number of jamming bits required to prevent communication. The methodology for which the number of jamming bits required is effectively minimised is the second jamming methodology in which the error patterns result in burst errors after decoding at the receiver, this jamming methodology requires only four jamming bits to be transmitted per CCCH block of 456 information bits, an improvement of 99.12% over an always on CCCH approach. The other methodologies tested also provided significant improvements over an always on jammer and all methodologies showed improvements over another recent control channel jamming strategy proposed in [18].

5.3 Conclusion

In this research we propose and explore four different jamming methodologies which exploit different parts of the control channel forward error correction scheme in use by the GSM system. In each case this is done in attempt to minimize the number of jamming bits required to prevent communications on the channel. Each methodology explored is capable of significantly reducing the number of jamming bits required, with the "errors in bursts" jamming approach effectively minimizing the number of jamming bits required to only four bits per 456 bit CCCH message.

5.4 Recommendations for Possible Future Work

For future works recommendations include research into more efficient methods of jamming current calls, without requiring the initial 5.296s constant jamming time to end current connections. Research can also be done into methods of making the jammer more specialised by targeting specific information elements, so it can target specific users, or allow specific numbers to be called such as emergency services.

References

[1] D. Chambers, “Mobile Network Statistics for 2016,” ThinkSmallCell, 18 02 2016. [Online]. Available: https://www.thinksmallcell.com/Opinion/mobile-network-statistics-for-2016.html. [Accessed 15 01 2017].

[2] C. Miller, “Cell Phone Bombs,” Cgynus Law Enforcement Group, 13 12 2006. [Online]. Available: http://www.officer.com/article/10250461/cell-phone-bombs. [Accessed 30 08 2015].

[3] A. Hussain and A. Saqib, “Protocol Aware Shot-Noise based Radio Frequency Jamming Method in 802.11 Networks,” in *Wireless and Optical Communications Networks (WOCN)*, Paris, 2011.

[4] M. Wilhelm, I. Martinovic, J. B. Schmitt and V. Lenders, “Short Paper: Reactive Jamming in Wireless Networks How Realistic is the Threat?,” in *WiSec’11*, Hamburg, Germany., 2011.

[5] D. Schneider, “The silence of the cellphones,” *Spectrum IEEE,* vol. 46, no. 4, p. 14, 2009.

[6] D. Thuente and M. Acharya, “Intelligent Jamming in Wireless Networks with Applications to 802.11b and Other Networks,” in *MILCOM*, Washington D.C., 2006.

[7] J. A. S. G. Z. Anthony D. Wood, “DEEJAM: Defeating Energy-Efficient Jamming in IEEE 802.15.4-based Wireless Networks,” Department of Computer Science, University of Virginia.

[8] A. Brand and H. Aghvami, “Multiple Access Protocols for Mobile Communications,” in *GPRS, UMTS and Beyond*, England, John Wiley & Sons, Ltd, 2002, pp. 107,108.

[9] G. Association, “Bried History of GSM & the GSMA,” GSM Association, [Online]. Available: http://www.gsma.com/aboutus/history. [Accessed 26 08 2015].

[10] E. T. S. I. “GSM Technical Specification 05.05,” ETSI, 1996.

[11] Shri, “GSM: Physical & logical Channels,” Learn Telecom, 01 08 2011. [Online]. Available: http://learntelecom.com/gsm-physical-logical-channels/. [Accessed 24 08 2015].

[12] I. Poole, “GSM Frame Structure,” Radio Electronics, [Online]. Available: http://www.radio-electronics.com/info/cellulartelecomms/gsm_technical/frames-structure-super-hyper.php. [Accessed 25 04 2015].

[13] E. T. S. I. “GSM Technical Specification 05.02,” ETSI, 1996.

[14] Ericsson, “Channel Concept,” 21 06 2013. [Online]. Available: http://www.slideshare.net/TempusTelcosys/02-channel-concept. [Accessed 24 04 2015].

[15] “Dedicated Control Channel (DCCH) in GSM,” TELETOPIX.ORG, 14 06 2012. [Online]. Available: http://www.teletopix.org/gsm/dedicated-control-channel-dcch-in-gsm/. [Accessed 25 08 2015].

[16] D. Adamy, “EW 101,” in *A First Course in Electronic Warfare*, Massachusetts, Artech House, Inc, 2001, p. 177.

[17] J. Eberspächer, H.-J. Vögel, C. Bettstetter and C. Hartmann, “GSM - Architecture, Protocols and Services,” in *3rd Edition*, John Wiley & Sons Ltd, 2009, pp. 102-110.

[18] M. Petracca, M. Vari, F. Vatalaro and G. Lubello, “Performance Evaluation of GSM Robustness Against Smart Jamming Attacks,” in *5th International Symposium on Communications, Control and Signal Processing, ISCCSP*, Rome, 2012.

[19] E. Biglieri, in *Coding for Wireless Channels*, United States of America, Springer Science+Business Media, Inc, 2005, p. 11.

[20] S. Lin and D. J. Costello, “Error Control Coding,” in *Second Edition*, India, Pearson Education, 2010, pp. 292,1107,1108.

[21] J. L. Burbank, J. Andrusenko, J. S. Everett and W. T. Kasch, “Wireless Networking: Understanding Internetworking Challenges,” Piscataway, IEEE Press, 20.

[22] P. Ostergard, “Systematic Cyclic Codes,” [Online]. Available: http://www.comlab.hut.fi/studies/3410/slides_08_6_4.pdf. [Accessed 06 09 2015].

[23] P. T. Komiske, “Error Detection and Correction Codes,” *APL Technical Digest,* p. 11, 12 1965.

[24] E. T. S. I. “GSM Technical Specification 05.03,” ETSI, 1996.

[25] J. S. Berg, in *Broadcasting on the Short Waves, 1945 to Today*, North Carolina, McFarland & Company, Inc, 2008, p. 44.

[26] W. Xu, W. Trappe, Y. Zhang and T. Wood, “The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks,” in *The ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Illinois, 2005.

[27] A. Hussain, N. A. Saqib, U. Qamar, M. Zia and H. Mahmood, “Protocol-Aware Radio Frequency Jamming inWi-Fi and Commercial Wireless Networks,” *JOURNAL OF COMMUNICATIONS AND NETWORKS,* vol. 16, no. 4, pp. 397-406, 2014.

[28] W. Xu, W. Trappe and Y. Zhang, “Jamming Sensor Networks: Attack and Defense Strategies,” *IEEE Network,* vol. 20, no. 3, pp. 41-47, 2006.

[29] R. Stuhlfauth, “GSM and GPRS System Information,” ROHDE&SCHWARZ, Munich.

[30] D. Nguyen, C. Sahin, B. Shishkin, N. Kandasamy and K. R. Dandekar, “A real-time and protocol-aware reactive jamming framework built on software-defined radios,” *In Proceedings of the 2014 ACM workshop on Software radio implementation forum (SRIF '14),* pp. 15-22, 2014.

[31] M. Acharya, T. Sharma, D. Thuente and D. Sizemore, “Intelligent Jamming in 802.11b Wireless Networks,” in *OPNETWORK 2004*, August 2004.

[32] MathWorks, “Communications System Toolbox,” [Online]. Available: https://www.mathworks.com/products/communications.html. [Accessed 5 12 2016].

[33] “Government uses jammers often - expert,” Independant Online, 21 02 2015. [Online]. Available: http://www.iol.co.za/news/politics/government-uses-jammers-often-expert-1.1821728#.VeWnQa0-4wE. [Accessed 01 09 2015].

[34] W. Xu, W. Trappe, Y. Zhang and R. University, “Jamming sensor networks: attack and defense strategies,” *Network, IEEE,* vol. 20, no. 3, pp. 41-47, 2006.

[35] W. Shen, P. Ning, X. He and H. Dai, “Ally Friendly Jamming: How to Jam Your Enemy and Maintain Your Own Wireless Connectivity at the Same Time,” North Carolina.

[36] C. F. C. “CONSUMER ALERT: Using or Importing Jammers is Illegal,” 06 03 2012. [Online]. Available: https://apps.fcc.gov/edocs_public/attachmatch/DA-12-347A1.pdf. [Accessed 30 08 2015].

[37] “Communication Systems Operator,” Defence Careers, [Online]. Available: http://www.defencecareers.mil.nz/army/jobs/communication-systems-operator. [Accessed 21 07 2015].