NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

A DETAILED ANALYSIS AND OPTIMIZATION OF THE MODIFIED POLAR DECODING RNTI RECOVERY METHOD TO TRACK USER ACTIVITY IN 5G NETWORKS

by

Christopher J. Richards

September 2022

Thesis Advisor:	Murali Tummala
Co-Advisor:	John C. McEachen

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2022	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE A DETAILED ANALYSIS AND OPTIMIZATION OF THE MODIFIED POLAR DECODING RNTI RECOVERY METHOD TO TRACK USER ACTIVITY IN 5G NETWORKS	**5. FUNDING NUMBERS** RE404
6. AUTHOR(S) Christopher J. Richards	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

In this thesis, we analyze and optimize the modified polar decoding and syndrome matching radio network temporary identifier (RNTI) recovery method to de-anonymize the physical downlink control channel (PDCCH) in 5G networks. We present the impact on RNTI recovery of payload length, codeword length, signal-to-noise ratio (SNR) and the Hamming and longest common substring (LCS) recovery methods. Further, we consider the full set of RNTIs and downlink control information (DCI) fields that can be examined for user activity data and propose methods to track user activity within radio networks from the recovered data. Finally, we optimize the RNTI recovery method for different attacker scenarios to demonstrate how an attacker can recover RNTIs, track UEs, and aggregate data about the UE usage patterns and/or metadata about the user.

14. SUBJECT TERMS 5G, downlink control information, DCI, Hamming, longest common substring, LCS, physical downlink control channel, PDCCH, polar coding, radio network temporary identifier, RNTI, signal-to-noise ratio, SNR, user equipment, UE	**15. NUMBER OF PAGES** 159
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

A DETAILED ANALYSIS AND OPTIMIZATION OF THE MODIFIED POLAR DECODING RNTI RECOVERY METHOD TO TRACK USER ACTIVITY IN 5G NETWORKS

Christopher J. Richards
Lieutenant Commander, United States Navy
BS, Cornell University, 2008

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2022

Approved by: Murali Tummala
Advisor

John C. McEachen
Co-Advisor

Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In this thesis, we analyze and optimize the modified polar decoding and syndrome matching radio network temporary identifier (RNTI) recovery method to de-anonymize the physical downlink control channel (PDCCH) in 5G networks. We present the impact on RNTI recovery of payload length, codeword length, signal-to-noise ratio (SNR) and the Hamming and longest common substring (LCS) recovery methods. Further, we consider the full set of RNTIs and downlink control information (DCI) fields that can be examined for user activity data and propose methods to track user activity within radio networks from the recovered data. Finally, we optimize the RNTI recovery method for different attacker scenarios to demonstrate how an attacker can recover RNTIs, track UEs, and aggregate data about the UE usage patterns and/or metadata about the user.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

2G	Second-generation Telecommunications
3G	Third-generation Telecommunications
3GPP	Third-generation Partnership Project
4G	Fourth-generation Telecommunications
5G	Fifth-generation Telecommunications
AWGN	Additive White Gaussian Noise
BCCH	Broadcast Control Channel
BWP	Bandwidth Part
CBG	Code Block Group
CE	Control Element
CRC	Cyclic Redundancy Check
C-RNTI	Cell Radio Network Temporary Identifier
CSI	Channel State Information
CS-RNTI	Configured Scheduling Radio Network Temporary Identifier
dB	Decibel
DCI	Downlink Control Information
DMRS	Demodulation Reference Signal
eMBB	Enhanced Mobile Broadband
GF	Galois Field
gNB	gNodeB
GUTI	Globally Unique Temporary User Equipment Identity
HARQ	Hybrid Automatic Repeat Request
IMSI	International Mobile Subscriber Identifier
INT-RNTI	Interruption Radio Network Temporary Identifier
I-RNTI	Inactive Radio Network Temporary Identifier
IoT	Internet of Things
LCS	Longest Common Substring
LDPC	Low-density Parity-check
LLR	Log-Likelihood Ratio
MAC	Medium Access Control

MCS	Modulation and Coding Scheme
MCS-C-RNTI	Modulation and Coding Scheme Cell Radio Network Temporary Identifier
MSB	Most Significant Bit
MTC	Machine-type Communications
nID	Cell ID
NDI	New Data Indicator
NG-RAN	Next-generation Radio Access Network
NR	New Radio
PBCH	Physical Broadcast Channel
PDCCH	Physical Downlink Control Channel
PDF	Probability Distribution Function
PDSCH	Physical Downlink Shared Channel
PRACH	Physical Random-access Control Channel
PRB	Physical Resource Block
P-RNTI	Paging Radio Network Temporary Identifier
PUCCH	Physical Uplink Control Channel
PUSCH	Physical Uplink Shared Channel
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Key
RA-RNTI	Random-access Radio Network Temporary Identifier
RNTI	Radio Network Temporary Identifier
RV	Redundancy Version
SC	Successive Cancellation
SCL	Successive Cancellation List
SCS	Successive Cancellation Stack
SFI	Slot Format Indicator
SFI-RNTI	Slot Format Indicator Radio Network Temporary Identifier
SI-RNTI	System Information Radio Network Temporary Identifier
SNR	Signal-to-Noise Ratio
SP-CSI-RNTI	Semi-persistent Channel State Information Radio Network Temporary Identifier

SRS	Sounding Reference Signal
SS	Synchronization Signal
SUPI	Subscriber Unique Permanent Identifier
TA	Timing Advance
TB	Transport Block
TC-RNTI	Temporary Cell Radio Network Temporary Identifier
TMSI	Temporary Mobile Subscriber Identity
TPC	Transmit Power Control
TPC-PUSCH-RNTI	Transmit Power Control Physical Uplink Shared Channel Radio Network Temporary Identifier
TPC-PUCCH-RNTI	Transmit Power Control Physical Uplink Control Channel Radio Network Temporary Identifier
TPC-SRS-RNTI	Transmit Power Control Sounding Reference Signal Radio Network Temporary Identifier
UE	User Equipment
URLCC	Ultra-reliable Low Latency Communications
VoIP	Voice over Internet Protocol
VRB	Virtual Resource Block
XOR	Exclusive OR

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First, I would like to thank my wife, Carly, for her endless support and encouragement throughout this challenging and time-consuming endeavor. Second, I'd like to thank my advisors Professors John McEachen and Murali Tummala. Thank you for sharing your extensive subject knowledge, meticulous attention to detail, and persistent dedication from the exploration phase to the final draft. It's been a great experience working through this process at the fantastic learning environment of Naval Postgraduate School.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

At the end of 2021, the number of mobile subscriptions in the world reached an estimated 8.1 billion, with a total mobile traffic estimate of 65 exabytes/minute [1]. Today, almost 2 billion people have a mobile phone and while many utilize 2G through 4G networks, fifth-generation (5G) mobile subscriptions are expected to grow from 660 million in 2021 to 4.4 billion in 2027 [1]. This increase in mobile subscriptions includes IoT devices with 30.2 billion internet of things (IoT) device connections forecasted by 2027 [1]. With the ubiquity of mobile devices, many transported continually by users and others performing critical tasks, privacy and security have become more important than ever. The third-generation partnership (3GPP) standard for 5G mobile communications has made significant improvements in mobile security and privacy [2]. However, in [3] a methodology was demonstrated to de-anonymize the physical downlink control channel (PDCCH) and recover a sample 12-bit message. The objective of this work is to expand on [3] to determine the feasibility of recovering PDCCH messages in practice, optimize the recovery method, and reveal user activity based on the information recovered. This work is intended to inform the mobile telecommunications industry and standards organizations on the vulnerabilities of the PDCCH so that further improvements may be made to the privacy and security of next generation mobile telecommunications.

A. PRIVACY IN MOBILE COMMUNICATIONS

The transition in mobile standards from 2G through 5G have included significant improvements in privacy and security for the users of the networks [4], [5]. A key aspect of mobile security in 5G is privacy protection, which ensures information about user equipment (UE) does not become available to others [6]. 2G standards first introduced an identifier called the temporary mobile subscriber identifier (TMSI) to protect a user's international mobile subscriber identity (IMSI), which does not change and is associated with the user's mobile billing account [4]. The TMSI, however, had a vulnerability in that the UE passed the IMSI itself to authenticate itself to a network before receiving a TMSI and thus could be recovered and tracked by a prudent attacker [4]. Further, 2G standards did not protect signaling traffic and failed to provide a mechanism for the UE to authenticate the network prior to joining it [4]. The 3G security architecture improved on 2G by updating to stronger protection of signaling and reducing the requirement to send the IMSI in the authentication process [4]. 4G improved mechanisms to prevent attackers from learning mobile user identities by introducing the globally unique temporary user equipment identity (GUTI), imposing encryption on signaling, and scrambling message recovery with the radio network temporary identifier (RNTI) [5]. The RNTI provided a unique and temporary address for a UE connected to a cell, which changed every time a UE connected to a new cell [5]. 5G networks further improved security by upgrading the authentication process and introducing the subscription permanent identifier (SUPI) and subscription concealed identifier (SUCI) all while using RNTIs to locally control radio link control channel signaling [2], [6].

The transformation to 5G is built upon technological advances that improve how the mobile waveform is coded, scheduled, and transmitted, and further expands the operating spectrum to include millimeter wave frequencies [7]. These advances are advertised to establish new use cases built around ultra-reliable low latency communications (URLCC), massive machine-type communication (MMTC) for the IoT, and enhanced mobile broadband (eMBB) [7]. In all use cases, privacy and security are paramount. In a URLCC use case such as a smart city using 5G to direct vehicular traffic, the ability to identify a specific vehicle and track its movements can compromise the privacy and possibly the security of the vehicle and its occupants. In an industrial MMTC IoT use case, where 5G networks can be used in industries such as oil & gas refining, the location and communications of critical safety devices could be compromised, leading to physical or cyber attacks. Lastly, in a eMBB use case, which is expected to be heavily utilized by mobile users, UE RNTI values could be correlated with the geographical movements of an individual, leading to the individual being tracked as they move about or change applications within the 5G cell.

B. A DE-ANONYMIZATION ATTACK IN 5G

In 5G networks, a UE is connected to the cellular network over the air via a radio link to a gNodeB (gNB) [8]. Within the gNB to UE link, there exists a control channel called the physical downlink control channel (PDCCH), and this channel directs the UE when, where, and how it can find its downlink data and transmit its uplink data [8]. The PDCCH is essentially directing traffic on the uplink and downlink to maintain the connection between the gNB and UE so that texts, calls, and application data can seamlessly flow. While all channels carrying the user data are encrypted, the PDCCH is not encrypted but instead is scrambled by the UE temporary identifier for the control channel, the RNTI [9]. Every PDCCH message is broadcast to all recipients and every UE will attempt to descramble the received message; however, only the UE descrambling with its own RNTI will be able to recover the message details [9].

It has been shown in [3] that the lack of encryption presents a vulnerability as the scrambling associated with the RNTI can be identified through a process called modified polar decoding and syndrome matching [3]. While recovering the RNTI is not easy and requires somewhat significant computational resources, the RNTI can be identified after numerous iterations [3]. This RNTI recovery provides a path to de-anonymize the PDCCH channel, recover the downlink control information (DCI) messages, and reveal information about a user's patterns, activities, and even location changes within the mobile environment. A prudent attacker could track multiple devices across the network, logging when the gNB changes parameters related to the device physical location or a change in mode of operation, e.g., the device has started a voice-over-internet-protocol (VOIP) call.

C. THESIS OBJECTIVE

The objective of this thesis is to optimize the RNTI recovery method for the expected PDCCH conditions in real-world 5G radio access networks (RANs). If an attacker can efficiently recover RNTIs in a 5G network, they can descramble the DCI commands sent in the PDCCH and in many cases track user activity on the network. Therefore, in this thesis we assess a sophisticated attacker's ability to recover RNTIs and track user activity for different payload lengths, codeword lengths, and signal-to-noise-ratios that are expected to be utilized.

To accomplish this objective, we first develop strategies to identify probable DCI payload lengths in the PDCCH given probability of bit error constraints. Next, we evaluate an optimal threshold to meet RNTI recovery goals in different mobile environments. Further, we consider the impact on RNTI recovery of increased payload length, increased codeword length, high and low signal-to-noise ratio, and Hamming versus longest common substring (LCS) recovery methods. Finally, we consider the information recovered from the DCI messages descrambled with the recovered RNTIs and evaluate how the aforementioned factors impact our ability to track UE activity within the mobile environment.

D. RELATED WORK

In [3], Gardner developed a method to recover RNTIs from intercepted PDCCH messages using modified polar decoding and syndrome matching. This work demonstrated the RNTI recovery method for a payload of 12 bits and codeword lengths around 128 bits. In this thesis, we expand this method for payload and codeword lengths as expected in the real-world 5G networks and develop additional techniques for an approach to optimize RNTI recovery based on the mobile environment and attacker objectives.

Egilmez et. al [10] characterized the error correction and error detection performance of the polar codes in the 3GPP 5G standard. This work demonstrated the baseline probability of block error P_{bler} performance of a PDCCH polar code for different payload lengths A and provided observations on the viability of polar codes for future standards. We develop a methodology in this thesis to evaluate the P_{bler} of the PDCCH polar code to compare results and then evaluate the RNTI recovery vulnerability exhibited through polar coding to further appraise the viability of their use.

In [11], Garrett evaluated brute-force and known plaintext attacks against the physical downlink shared channel (PDSCH) which uses low-density parity-check (LDPC) codes for error correction. The work concluded that the LDPC coded PDSCH better protected the anonymity of an RNTI against such attacks than the polar coded PDCCH. In this thesis, we develop an attacker's methodology to optimally attack the PDCCH to provide further corroboration of the vulnerabilities of the polar coding used in the PDCCH.

E. THESIS OVERVIEW

The remainder of this thesis is structured as follows. Chapter II provides mobile communication background, a 5G channel overview, the technical details of the method to de-anonymize the PDCCH developed in [3], and the types of RNTIs and DCI messages used in the PDCCH as established by the 3GPP standards. Chapter III describes how to determine probable DCI payloads and optimize the methodology used for RNTI recovery to ultimately track user activity. Chapter IV presents the simulation setup and results of the DCI payload analysis and RNTI recovery methodology for practical payload and codeword combinations at different threshold values, signal-to-noise ratios (SNR), and matching methods. Chapter V concludes with the takeaways from the results and provides recommendations for future work in analyzing 5G physical channel vulnerabilities. Appendix A contains the MATLAB code for evaluating the DCI payload lengths within probability of bit error constraints. Appendix B contains the MATLAB code to recover RNTIs in the blind from scrambled PDCCH messages.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

In this chapter, we provide the foundation critical to understanding the methodology to recover RNTIs and track user activity information from intercepted PDCCH messages. An overview of mobile communication metrics is provided to develop basic concepts with which we can analyze changes in the mobile environment. This is followed by an overview of 5G physical channels to provide an understanding of the importance of the PDCCH in controlling traffic in 5G networks. Next, we present a method developed in [3] to use modified polar decoding and syndrome matching to recover RNTIs in the blind from intercepted PDCCH messages. Finally, a detailed description of the RNTIs and DCI messages used in 5G is provided as these identifiers and messages can be used to recover user activity from the PDCCH.

A. MOBILE COMMUNICATION METRICS

In this thesis, we assess the activity of a user based on the changes to the encoded information recovered from the intercepted PDCCH messages. This section provides a brief introduction to important concepts in mobile communication, which are critical to translating between physical changes in the channel and technical information gleamed from the PDCCH.

1. Wireless Communication Link Measurements

The first wireless communication concept to understand is that of received power P_R in a communications link,

$$P_R = \frac{P_T G_T G_R}{L_C} \tag{2.1}$$

where P_T is the transmitted power, G_T is the transmitter gain, G_R is the receiver gain, and L_C is the path loss [12]. Further,

$$L_C = \left(\frac{4\pi d}{\lambda} \right)^2 \tag{2.2}$$

where d is the distance between the transmitter and reciever, and λ is the wavelength of the communications link [12]. Cellular communications frequently operate with multipath channels, but in general the relations of (2.1) and (2.2) can be used to estimate whether P_R increases or decreases as distance between the UE and gNB changes [12]. The takeaway for analyzing PDCCH messages is that when a UE moves further from a gNB, P_R will decrease and can result in a request from the gNB for an increase in P_T.

Second, it is important to understand the probability of channel bit error $P_{b,QPSK}$ in the PDCCH. The PDCCH uses coherently detected quadrature phase shift key (QPSK) to modulate the bits onto the carrier waveform for transmission [9]. The probability of bit error for coherently detected QPSK in an additive white Gaussian noise (AWGN) channel is well established using the Q-function as

$$P_{b,QPSK} = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{2.3}$$

where E_b is the energy per bit, and N_0 is the noise spectral density [13]. In the PDCCH, as E_b decreases or N_0 increases, $P_{b,QPSK}$ will increase since the Q-function output decreases as its positive argument increases. To keep $P_{b,QPSK}$ constant, this increase could be counteracted by either driving an increase in E_b or adding more error correction bits as will be discussed in the next section. In mobile communications, due to high channel bit error probabilities, error correction coding is typically applied to improve the information bit error P_b; however, as $P_{b,QPSK}$ increases or decreases, P_b will as well.

Lastly, the signal-to-noise ratio (SNR) is an important metric to understand the relationship between the gNB, the UE, and an attacker's intercepting location. The SNR, typically represented in dB, is defined by

$$SNR = \frac{P_R}{P_N} = \frac{E_b R_b}{N_0 B} \tag{2.4}$$

where P_R is the received signal power, P_N is the noise power, E_b is the energy per bit, R_b is the bit rate, N_0 is the noise spectral density, and B is the signal bandwidth [12]. It is important to note that an attacker passively monitoring the PDCCH will likely experience a SNR different from that of the UE for which the communications are intended.

2. Error Detection, Correction, and Polar Coding

Central to this thesis are the concepts of error detection and error correction in which errors in information bits can be respectively detected and corrected using code bits. The first method we introduce is the calculation of the cyclic redundancy code (CRC), which performs error detection only. In the PDCCH, a CRC is used to ensure the received data bits, which carry a DCI message are not corrupted [9]. The CRC is calculated by taking the cyclic generator polynomial

$$g_{24C} = x^{24} + x^{23} + x^{21} + x^{20} + x^{17} + x^{15} + x^{13} + x^{12} + x^{8} + x^{4} + x^{2} + x + 1 \tag{2.5}$$

where x denotes a bit delay of a length equal to its exponent, and dividing g_{24C} by the data bits in Galois Field (GF)(2) [9]. The length 24-bit CRC is then appended to the length A data bits to form a length K block [9]. This CRC, when calculated can determine whether the data bits have been corrupted from the originally sent bits, or, as we will apply in this thesis, when the data bits are uncorrupted and have been descrambled with the correct scrambling sequence [3].

While error detection is used to confirm the A data bits have been accurately received, it is necessary to add bits for error correction to reduce the information bit error probability, P_b, as mobile communication channels are known to have high channel error probabilities. The PDCCH uses block error correction by means of polar coding, where K bits (data + CRC) are encoded onto a block of N polar coded bits [14]. In polar coding, first $F = N - K$ zeros are inserted, referred to as frozen bits to generate a codeword of length N [3]. The sequence locations in which the F bits are inserted are pre-determined to place the K bits in the most reliable positions, i.e., in positions most likely to be corrected if an error occurs in transmission [3], [14]. Once the length N polar coded block is formed, the polar coding bits are generated by

$$p_0^{N-1} = n_0^{N-1} G_N \tag{2.6}$$

where p_0^{N-1} are the polar coded bits, n_0^{N-1} are the input bits to polar coding, and G_N is the polarization matrix. It follows that

$$G_N = G_2^{\otimes n} \tag{2.7}$$

where $\otimes$ is the Kronecker product, and multiplication is performed in GF(2) and

$$G_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \tag{2.8}$$

is the polarization matrix when $N = 2$ [3], [14]. The end effect of the GF(2) polarization matrix multiplication is that the K data + CRC bits are diffused throughout the N polar coded bits so that when bit errors occur in transmission, it is possible to correct the errors and fully recover the K bits [14]. The important concept to note here is that an increase in F results in an increase in error correction capabilities, also referred to as code rate, r, which ultimately decreases the information bit error probability, P_b. Throughout this thesis we will vary payload length A and codeword length E, both of which directly impact the size of F. For further background on polar coding and its error correction capabilities, the reader is referred to [3], [10], and [14].

Polar decoding in this thesis follows [3] and uses log likelihood ratio (LLR) based successive cancellation (SC) decoding to recover the K data + CRC bits from the length N polar coded block. In this case, the received QPSK symbols are translated to LLR values instead of hard 0 and 1 bits, which in the end improves error correction [3], [12]. The polar decoding process, detailed in [3] and [12], inputs the LLR values and corrects errors in the received bits to produce the recovered K data + CRC bits. The power of the polar decoding is directly related to the number of frozen bits F, which is a key concept in this thesis. In addition, there are more complex methods to perform polar decoding, such as successive cancellation list (SCL) decoding and successive cancellation stack (SCS) decoding, to improve the error correction capabilities requiring increased computational power; more information can be found in [10], [15], [16], and [20].

B. PHYSICAL CHANNELS IN 5G NETWORKS

To understand the methodology used to recover RNTIs and decode DCI information informing user activity, an overview of 5G communications channels is provided. Specifically, it is imperative to understand the process to encode and decode PDCCH messages as in this process is where a vulnerability to recover RNTIs lies. A brief introduction to the method to recover RNTIs is provided in this section as well as an overview of all the potentially recoverable user activity information.

1. Downlink and Uplink Physical Channels

The physical downlink channels in 5G networks consist of the physical broadcast channel (PBCH), physical downlink control channel (PDCCH), and the physical downlink shared channel (PDSCH) [18]. The PBCH broadcasts key information that a UE requires to access the cell and is one of the first channels received when a UE attempts to connect to a 5G network [17]. Once a UE is connected and authenticated, the PDDCH is used by the gNB to transfer downlink configuration information via DCI messages [17]. These DCI messages contain critical configuration information in which a UE receives its downlink resource allocation so that it may transmit to the gNB and its uplink resource allocation so that it may receive data transmissions from the gNB [17]. The downlink resource allocation provides access to the PDSCH, a shared channel on which the UE can receive application data, signaling messages, system information messages, paging messages, and some control information [17].

On the uplink side, the physical channels are the physical random-access channel (PRACH), the physical uplink control channel (PUCCH), and the physical uplink shared channel (PUSCH) [18]. The PRACH is used by UEs connecting to the network to send messages as required by the random-access procedure, which governs the process for a UE to initially connect and authenticate to a network [17]. Once connected, the PUCCH is used to transfer uplink configuration information via PUCCH format messages. These messages are similar to DCI messages and control hybrid automatic repeat request (HARQ) acknowledgments, scheduling requests, and channel state information (CSI) reports from the UE. We note here that the PUCCH transmits information about the channel conditions and similarly is polar coded and scrambled by a RNTI sequence, thus making it vulnerable to the RNTI recovery methods discussed in this thesis. However, this thesis is limited in scope to the PDCCH, chosen because it contains more valuable activity information than the PUCCH. Finally, the PUSCH is a shared channel used to transfer application data, signaling messages, and some control information [17]. The PUSCH resource allocation is sent on the PDCCH, which is the focus of this thesis [17].

2. PDCCH Encoding and Modulation

To understand the nuances of the RNTI recovery process presented in [3], the PDCCH modulation and coding process is detailed in this section. As shown in Figure 1, data to be sent in the PDCCH is modified by scrambling, interleaving, polar encoding, and rate matching processes prior to transmission, where all steps are governed by [18], a 3GPP standard. The first step of note to this thesis is the CRC scrambling, which as discussed in Section II.A.2 provides a method to confirm whether all errors have been corrected in the data bits and for our purposes whether the data bits have been descrambled correctly if the RNTI is not known *a priori* [3]. Second, the frozen bit insertion and polar coding, as discussed in Section II.A.2, is the step in which the frozen bits are inserted, and the polar coding method is applied. These two steps are critical to this thesis in that the impact of the scrambling sequence and subsequent polar coding on the frozen bits will be analyzed to recover the RNTI. The rate matching step which follows polar coding is in place to adjust the polar coded block length N to the codeword length E, which is assigned by the gNB according to the resources available [17]. In this step, if necessary, the length N block of polar coded bits undergoes either repetition, shortening, or puncturing to be adjusted to length E [9]. This thesis does not go into detail on the rate matching process, but the reader can find the background in [9] and further explanation and examples in [3].

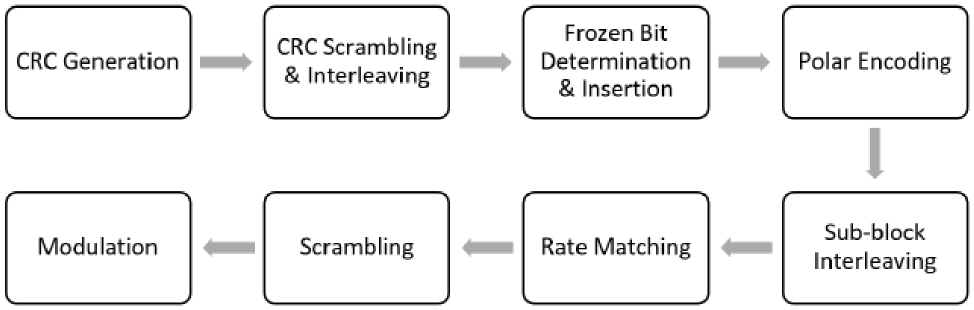


Figure 1. Physical Downlink Control Channel Modulation and Coding Process. Source: [3], [9].

Next, it is important to note that the scrambling step uses a length-31 Gold sequence initialized by a generator according to

$$c_{init} = (n_{RNTI} \cdot 2^{16} + \eta_{ID}) \bmod 2^{31} \tag{2.9}$$

where c_{init} is the scrambling sequence initiator, n_{RNTI} is the RNTI assigned, and η_{ID} is the assigned cell ID [18]. The $\bmod 2^{31}$ discards the 16th bit from the RNTI, thus reducing our scrambling sequence space to 2^{15} possibilities [3]. Once the rate matched bits of length E are scrambled, the resultant bits are modulated and transmitted as QPSK symbols [18]. From the perspective of an attacker analyzing an intercepted PDCCH message, it is important to note that while the output codeword length E will be recovered, the input data length A will not be known and this length affects the frozen bit determination, frozen bit insertion and rate matching steps in the encoding process. Further for the attacker, there is no *a priori* knowledge of the RNTI, which determines the scrambling sequence used.

3. PDCCH Demodulation and Decoding

The PDCCH demodulation and decoding process, as shown in Figure 2, is essentially the reverse of the encoding and modulation steps presented in the previous section; for the UE, the possible lengths of the DCI message A and the RNTI will be known [18]. A process called DCI size alignment is performed for the PDCCH in which the gNB has configured its DCI messages to be no more than four sizes [9]. This is accomplished through padding as explained in more detail in Section II.D.2, but for our purposes it means that there are four possible values of A. The UE decoding the PDCCH bits received will attempt to perform the DCI message recovery process for one of the four sizes and if the CRC does not match the decoded bits, the UE will attempt to decode the next size until the message is recovered or all four sizes are exhausted [9].

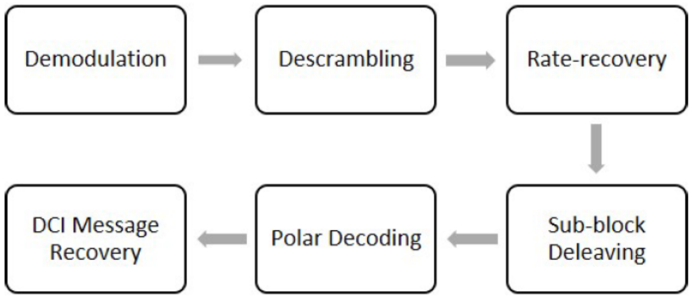


Figure 2. Physical Downlink Control Channel Demodulation and Decoding Process. Source: [3], [9].

In this case, the demodulation is performed using LLR SC decoding, the RNTI initiated sequence is applied to descramble the bits, and the remaining steps taken to encode the message are reversed to recover the DCI message and the CRC [3], [9]. At this point, the UE calculates a CRC for the received A data bits and compares it to the CRC decoded within the K bits recovered [9]. If the CRC matches, then the data bits have been successfully demodulated, descrambled, and decoded; if not, then there are three possibilities. The first possibility is that the RNTI is not correct, and this message is addressed to another UE. The second possibility is that the message length A is not correct for this message, and the UE will attempt the process using the other three sizes. The last possibility is that uncorrectable errors have occurred, and the message is unrecoverable. We see through this process how the PDCCH uses its RNTI as an address in that every UE will attempt to recover every PDCCH message but will only be able to descramble the message if it has been scrambled by its assigned RNTI [3], [9]. The other takeaway here is that every different A results in a different decoding sequence; therefore, for an attacker, knowing the four possible A values or having some idea of what they may be is critical.

C. METHOD TO RECOVER RNTIS

So far, we have covered the PDCCH encoding, modulation, demodulation, and decoding processes as designed in 5G networks. Now, we will cover how to exploit a vulnerability [3] in which modified polar decoding and syndrome matching can be used to decode DCI payloads from intercepted PDCCH messages without *a priori* knowledge of the RNTI. The methodology used in this section is detailed extensively with proof and examples in [3]; the reader is directed there for further information if desired.

1. Modified Polar Decoding

In [3], modified polar decoding was demonstrated as a method to identify the effect of the RNTI initiated scrambling sequence on the frozen bits inserted for polar coding. As shown in Figure 3, this method takes the LLR values recovered from the demodulation, rate recovery, and sub-block deinterleaving steps and performs modified polar decoding on those bits. To perform modified polar decoding, the recovered bit sequence is multiplied by the polarization matrix to reverse the forward polar coding process applied in the PDCCH encoding steps by

$$\hat{n}_0^{N-1} = \hat{p}_0^{N-1} G_N \tag{2.10}$$

where $\hat{p}_0^{N-1}$ are the polar coded bits estimated from the received LLR values, $\hat{n}_0^{N-1}$ are the received bits in which the polar coding process has been reversed, and G_N is the polarization matrix. From the modified polar decoding step output, we can take $\hat{n}_0^{N-1}$ and select only the frozen bits to form an error pattern of length, $\varepsilon = E - K$ [3].

In the case in which the RNTI is known, and descrambling is applied prior to polar decoding, the error pattern consists of all frozen bits and thus will consist of all zeros except when those zeros have been flipped due to additive white Gaussian noise (AWGN). In our case of modified polar decoding, the error pattern is uniquely affected by the RNTI scrambling sequence, considered as a form of non-random noise [3]. This pattern is still affected by AWGN and therefore some bits will be further flipped due to the AWGN. The takeaway of modified polar decoding is that it leaves us with an error pattern ε that is affected by the RNTI scrambling sequence and AWGN.

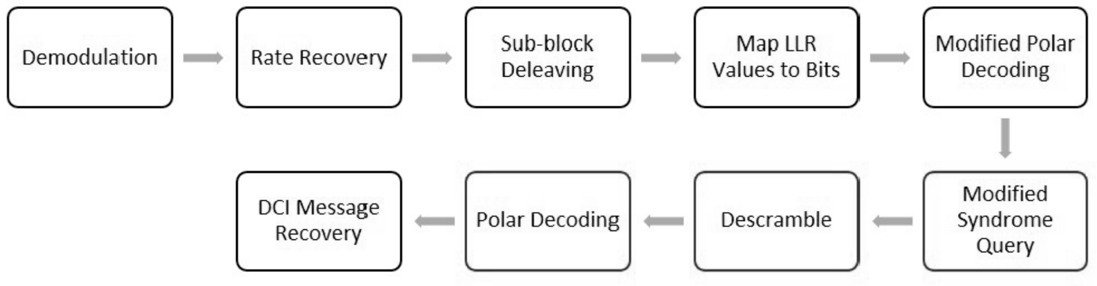


Figure 3. Modified Polar Decoding and Syndrome Matching Process to Recover RNTIs and Decode DCI Messages. Source: [3].

2. Syndrome Matching for RNTI Recovery

The method for determining the RNTI from the error pattern developed in [3] is to pre-generate a syndrome table, which consists of all possible RNTI scrambled error patterns and then to query the syndrome table for an error pattern match. Recall that there are 2^{15} possible RNTI initiated scrambling sequences; therefore, each syndrome table has 2^{15} syndrome entries [3]. However, each codeword and payload combination generates a different error pattern due to the number of frozen bits added and the unique sequence in which the frozen bits are added [3]. As a result, many syndrome tables are generated; however, once a syndrome table for a given codeword length E and payload length A is created, it can be used in perpetuity [3]. There is some complexity to generating syndrome tables for different rate-matching cases, but a method to develop the tables is presented in [3], and for the purposes of this thesis, we can generate a syndrome table for any E and A combination.

We must also consider that some of the error pattern bits will be flipped due to the presence of AWGN, which is a major focus of this thesis. Since the intercepted PDCCH sequence is affected by AWGN, the attacker cannot rely on a direct match but instead must apply a threshold within which an error pattern and syndrome can be considered a match [3]. The LCS method and the Hamming method can be used to filter matches to efficiently recover the RNTI [3]. The Hamming method considers the Hamming distance between the two sequences d_{HAM} and is set as a maximum threshold τ_{HAM} [3]. For a 10-bit example with a randomly generated error pattern and syndrome,

$$e_0^{\varepsilon-1} \oplus s_0^{\varepsilon-1} = \varphi_{HAM,0}^{\varepsilon-1} \tag{2.11}$$

$$[0\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0] \oplus [1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0] = [1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 0]$$

where $e_0^{\varepsilon-1}$ is the error pattern, $s_0^{\varepsilon-1}$ is the syndrome, $\phi_{HAM,0}^{\varepsilon-1}$ is the exclusive OR (XOR) result of the two, and ε is the length of the sequences. We find that $d_{HAM} = \sum_{i=0}^{\varepsilon-1} \varphi(i) = 4$, and if $d_{HAM} \le \tau_{HAM}$, then the error pattern and syndrome will be considered a match. A matching error pattern is then descrambled, and the CRC is checked to determine if the RNTI of that syndrome is the correct RNTI.

Another method to compare the error pattern and the syndrome is the LCS method, which considers the longest common substring d_{LCS} of matching bits between the two sequences and is set as a minimum threshold τ_{LCS} [3]. For the same 10-bit randomly generated example,

$$e_0^{\varepsilon-1} = [0\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0]$$

$$s_0^{\varepsilon-1} = [1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0]$$

we find that $d_{LCS} = 3$ as the last three bits of the sequence match, which is the longest common consecutive substring. Similar to the Hamming method, if $d_{LCS} \ge \tau_{LCS}$, the matching error pattern is then descrambled, and the CRC is checked to determine if the RNTI of that syndrome is the correct RNTI. The last item we consider is that there are 2^{15} possible RNTI initiated scrambling sequences, but 2^{16} RNTIs that could be associated with a UE. This problem is solved by descrambling with both the recovered RNTI n_{RNTI} and the identical 2^{15} sequence but with a 16th MSB bit of 1, $n_{RNTI} + 2^{15}$ [3]. Descrambling and decoding with both the RNTI values and checking for a correct CRC ensures that the entire RNTI search space is exhausted.

D. TYPES OF RNTIS AND DCI MESSAGES IN THE 5G PDCCH

In this section, we provide an overview of the RNTIs and DCI formats used in 5G networks as governed by the 3GPP NR standards [9], [18], and [19]. The types of RNTIs that are used is important because, as we recover these RNTIs, we can identify what type of activity is being directed by the gNB. Further, the DCI formats are uniquely scrambled by different RNTIs and contain DCI fields with valuable information that can be used to assess user activity.

1. RNTIs in the PDCCH

RNTIs are used as a unique address for DCI messages sent to specific UEs in the PDCCH [17], [19]. When PDCCH message is received by a UE, the UE does not know whether that message is intended for it or another UE and will demodulate the signal and begin attempting to decode the message with its own RNTI in the descrambling step [20]. The polar decoding process is designed such that a UE will quickly be able to identify whether its RNTI-initiated descrambling is correct and can abandon the process early if not, saving resources and battery life [20]. However, if the RNTI descrambling is correct, then the PDCCH message is descrambled, the DCI payload is retrieved, and the UE has received its message [20]. Overall, the 3GPP standard for NR uses thirteen unique RNTIs at the gNB level, and they are listed in Table 1. There is one RNTI used at the next-generation radio access network (NG-RAN) level to track inactive UEs called the inactive RNTI (I-RNTI), but since I-RNTI is not used to scramble PDCCH bits in the physical layer, it is not further discussed in this thesis [8].

Table 1. Types of RNTIs Used in the PDCCH. Source: [17], [19].

RNTI	DCI Format	Application	Value
SI-RNTI	1_0	PDSCH resources for System Information	FFFF
P-RNTI	1_0	PDSCH resources for Paging Messages	FFFE
RA-RNTI	1_0	PDSCH resources for Random-access Response (RAR)	Random (0001-FFEF)
TC-RNTI	0_0, 1_0	PDSCH resources for MSG3 re-transmissions, PDSCH resources for MSG4	
C-RNTI	0_0, 1_0, 1_0, 1_1	PUSCH and PDSCH resources for application data and control plane signaling	
MCS-C-RNTI	0_0, 1_0, 1_0, 1_1	Dynamic selection of low Spectral Efficiency MCS Table for PDSCH and PUSCH	
CS-RNTI	0_0, 1_0, 1_0, 1_1	Configured Grant Scheduling for PUSCH, Semi-Persistent Scheduling for PDSCH	
TPC-PUSCH-RNTI	2_2	Closed loop uplink power control commands for PUSCH	
TPC-PUCCH-RNTI	2_2	Closed loop uplink power control commands for PUCCH	
TPC-SRS-RNTI	2_3	Closed loop uplink power control commands for the SRS	
INT-RNTI	2_1	Interruption signaled using Pre-emption Indications	
SFI-RNTI	2_0	Dynamic changes to the slot format signaled using Slot Format Indicators	
SP-CSI-RNTI	0_1	Trigger to activate/deactivate Semi-Persistent CSI reporting from the UE	

Considering the RNTIs in Table 1, we first point out the fact that the SI-RNTI and P-RNTI have fixed hexadecimal values of FFFF and FFFE, respectively. These RNTIs are not unique to a UE and are used to scramble system information and paging messages which all UEs receive; therefore, they are not of interest to our user activity tracking methodology [8]. The random-access RNTI (RA-RNTI) and the temporary cell RNTI (TC-RNTI) are used in the random-access procedure to assign an identifier to a UE first connecting, or in some cases reconnecting, to a network [21]. The RA-RNTI is initially assigned based on calculated cellular parameters, and then the gNB assigns a TC-RNTI to the UE to release the RA-RNTI back for another UE attempting to join [21]. Frequently the TC-RNTI is then promoted to cell RNTI (C-RNTI) once the UE is authenticated [21].

The C-RNTI is the primary RNTI assigned to a UE for the gNB to dynamically allocate resources on the PDCCH [8]. While the use of the C-RNTI itself does not indicate the use of any special features, we considered it as the primary means to track UEs on mobile networks in this thesis.

Next, we consider the modulation and coding scheme cell RNTI (MCS-C-RNTI), the configured scheduling RNTI (CS-RNTI), and the semi-persistent channel state information RNTI (SP-CSI-RNTI). These RNTIs are used to activate a specific mode of operation in a UE of which the UE has been pre-configured to expect [8]. The use of the MCS-C-RNTI triggers dynamic MCS changes, the CS-RNTI triggers semi-persistent scheduling in the downlink or configured grant in the uplink, and the SP-CSI-RNTI triggers the use of semi-persistent CSI reporting on the PUSCH [8]. The identification of these RNTIs is useful in this thesis because they indicate possible changes in mode of operation that may be associated with a specific application or state of the UE. The RNTIs considered so far are used to scramble DCI format 0_0, 0_1, 1_0, and 1_1 messages, which direct scheduling of resources on the PUSCH and PDSCH [19].

Further, we consider the RNTIs used to scramble DCI format 2_0, 2_1, 2_2, and 2_3 messages, which are typically addressed to groups of UEs [9]. In these cases, the group of UEs would share the RNTI used to scramble these messages; within the DCI message, each UE would have a field reserved for its UE-specific command [9]. The transmit power control PUSCH RNTI (TPC-PUSCH-RNTI) and transmit power control PUCCH RNTI (TPC-PUCCH-RNTI) are used to scramble DCI 2_2 format messages to direct closed loop power control commands on the PUSCH and PUCCH, respectively [9]. Similarly, the transmit power control sounding reference signal RNTI (TPC-SRS-RNTI) is used to scramble DCI 2_3 format messages and direct closed loop power control commands for the SRS [9]. The INT-RNTI, used to scramble DCI 2_1 format messages, is used to indicate pre-emption in the downlink, which indicates to the group that its transmissions are being interrupted for time critical transmissions from another UE, which could be valuable information for our UE activity tracking [8]. The last RNTI in the PDCCH is the SFI-RNTI, used to scramble DCI format 2_0 and to direct a change to slot format in which the gNB can dynamically reconfigure resources for optimal use [8], [17]. The use of these RNTIs to deduce user activity and track changes to user activity will be discussed in Section III.D.1.

2. DCI Messages in the PDCCH

In this section, we will describe the DCI formats and the information contained within each as governed by the 3GPP NR standards. These DCI payloads when encoded are scrambled by the aforementioned RNTIs; therefore, recovering a RNTI for an intercepted PDCCH message allows for the DCI payload to be unscrambled. The DCI messages contain valuable configuration information for the PUSCH and PDSCH and identify special features activated [17]. Table 2 presents the types of DCI formats used in the PDCCH. The 'Fallback' messages are designed to maintain a connection when coverage deteriorates as they do not have as many configurable fields as the 'Standard' messages and are therefore smaller payloads, allowing for higher code rates to be used [17]. DCI formats 0_0 and 0_1 are used to dynamically schedule resource allocations on the PUSCH [9]. DCI formats 1_0 and 1_1 are used to dynamically schedule resource allocations on the PDSCH [9]. The DCI formats designed to support group messaging are DCI format 2_0, which notifies UEs of the slot format, DCI 2_1, which notifies UEs of a pre-emption for time critical transmission, DCI 2_2, which is used to transmit TPC commands for PUCCH and PUSCH, and DCI 2_3, which is used to transmit TPC commands for SRS [9].

Table 2. Types of DCI Messages Used in the PDCCH. Source: [9], [17].

DCI Format	Application
0_0	'Fallback' DCI format for uplink resource allocations on PUSCH
0_1	'Standard' DCI format for uplink resource allocations on PUSCH
1_0	'Fallback' DCI formats for downlink resource allocations on PDSCH
1_1	'Standard' DCI format for downlink resource allocations on PDSCH
2_0	Provision of Slot Format Indicators (SFI)
2_1	Provision of Pre-emption Indications
2_2	Provision of closed loop power control commands applicable to PUCCH and PUSCH
2_3	Provision of closed loop power control commands applicable to SRS

a. DCI 0_0 and 0_1 PUSCH Resource Allocations

The PUSCH allocations as directed by DCI formats 0_0 and 0_1 tell the UE how to operate when sending information to the gNB on the PUSCH [17]. First, we will consider the 'Fallback' format, DCI 0_0, and the comprehensive list of DCI fields for this format as shown in Table 3. The UE will be assigned specific time and frequency resources to use through the frequency domain resource assignment and time domain resource assignment fields [17]. While this uplink data on the PUSCH will be encrypted, this information can reveal when the UE is active on the channel. Further, the MCS, New Data Indicator, HARQ process number, and TPC Command for Scheduled PUSCH fields all contain information that can be used to inform the strength of the connection, which can be used to reveal information about UE activity, which will be discussed further in Section III.D.2. The uplink/supplemental uplink indicator field can additionally be used to inform the number of uplinks that the UE is active on [17]. Table 4 shows the comprehensive list of DCI fields for the 'Standard' DCI format 0_1, which as can be seen contains many more configurable options. In addition to the DCI fields listed above, the SRS Resource Indicator, SRS request, and CSI request fields could reveal additional information about the strength of the connection. Lastly, the carrier indicator field can indicate that the UE is configured across multiple carriers [17].

Table 3. DCI 0_0 PUSCH Resource Allocations (Fallback). Source: [9], [17].

Field	Brief Description of Field Contents
Identifier for DCI Format	Differentiates DCI format 0_0 and 1_0
Frequency Domain Resource Assignment	Allocates a source of resource blocks in the frequency domain
Time Domain Resource Assignment	Determines slot offset, PUSCH Mapping Type, starting symbol and number of allocated symbols
Frequency Hopping Flag	Indicates whether frequency hopping is applied
Modulation and Coding Scheme (MCS)	Defines modulation and coding scheme via 3GPP lookup table
New Data Indicator	Indicates if resource allocation is for a re-transmission
Redundancy Version	Indicates the puncturing pattern after channel coding
HARQ Process Number	Indicates the HARQ process to use the resource allocation
TPC Command for Scheduled PUSCH	Used for closed loop power control as UE is directed to increase, decrease, or maintain power
Padding	Added to DCI 0_0 to match size with DCI 1_0 to minimize UE blind decoding attempts
Uplink/Supplemental Uplink Indicator	Indicates whether resource allocation is for the normal uplink carrier or the supplemental uplink carrier

Table 4. DCI 0_1 PUSCH Resource Allocations (Standard). Source: [9], [17].

Field	Brief Description of Field Contents
Identifier for DCI Format	Differentiates DCI format 0_1 and 1_1
Carrier Indicator	Configures cross carrier scheduling between cells
Uplink/Supplemental Uplink Indicator	Indicates whether resource allocation is for normal uplink carrier or supplemental uplink carrier
Bandwidth Part Indicator	Identifies BWP for frequency domain resource allocation
Frequency Domain Resource Assignment	Allocates a source of resource blocks in the frequency domain
Time Domain Resource Assignment	Determines slot offset, PUSCH Mapping Type, starting symbol and number of allocated symbols
Frequency Hopping Flag	Indicates whether frequency hopping is to be applied
Modulation and Coding Scheme	Defines modulation and coding scheme via 3GPP lookup table
New Data Indicator	Indicates if resource allocation is for a re-transmission
Redundancy Version	Indicates the puncturing pattern after channel coding
HARQ Process Number	Indicates the HARQ process to use the resource allocation
1st Downlink Index	HARQ acknowledgment procedure for downlink data
2nd Downlink Index	HARQ acknowledgment procedure for downlink data
TPC Command for Scheduled PUSCH	Used for closed loop power control as UE is directed to increase, decrease, or maintain power
SRS Resource Indicator	Used to select SRS resources
Precoding Information & Number of Layers	If codebook based, selects Transmitted Precoded Matrix Indicator and number of layers
Antenna Ports	Indicates which logical antenna ports the UE should use
SRS Request	Triggers SRS Resource Sets configured for aperiodic trigger
CSI Request	Selects CSI "Trigger State" for aperiodic CSI trigger state
CBG Transmission Information	Used to select Code Block Groups to transmit uplink data
PTRS-DMRS	Links a Phase Tracking Reference Signal (PTRS) to a Demodulation Reference Signal (DMRS)
Beta Offset Indicator	Configures weights to be applied during the rate matching of uplink control information on the PUSCH
DMRS Sequence Indicator	If transfer precoding disabled, initializes pseudorandom sequence which populates DMRS Resource Elements
UL-SCH Indicator	Indicates if UL-SCH transmitted on the PUSCH
Padding	Added to match DCI 0_1 size if UE configured for both Supplemental and Normal uplink

b. DCI 1_0 and 1_1 PDSCH Resource Allocations

The PDSCH allocations as directed by DCI formats 1_0 and 1_1 tell the UE how to operate when receiving information from the gNB on the PDSCH [4]. First, we will consider the 'Fallback' format, DCI 1_0. The comprehensive list of DCI fields for this format is shown in Table 5. Similar to the PUSCH, the UE will be assigned specific time and frequency resources to use through the frequency domain resource assignment and time domain resource assignment fields, which can indicate when the UE is active [4]. In this format, the MCS, New Data Indicator, HARQ process number, TPC Command for Scheduled PUCCH, PUCCH to HARQ Feedback Timing Indicator, and Random-access Preamble Index fields all contain information that can be used to inform the strength of the connection. These can also be used to reveal information about UE activity, which will be discussed further in Section III.D.2. The uplink/supplemental uplink indicator can additionally be used to inform the number of uplink that the UE is active on. Table 6 shows the comprehensive list of DCI fields for the 'Standard' DCI format 1_1, which again contains many more configurable options. In addition to the DCI fields listed above, the SRS request field could reveal additional information about the strength of connection.

Table 5. DCI 1_0 PDSCH Resource Allocations (Fallback). Source: [9], [17].

Field	Description of Field Contents
Identifier for DCI Format	Differentiates DCI format 0_0 and 1_0
Short Message Indicator	Differentiates paging only or paging and scheduling
Short Messages	Informs UE with message regarding the BCCH
Frequency Domain Resource Assignment	Allocates a source of resource blocks in the frequency domain
Time Domain Resource Assignment	Determines slot offset, PUSCH Mapping Type, starting symbol and number of allocated symbols
VRB-to-PRB Mapping	Indicates if interleaving is used on PDSCH
Modulation and Coding Scheme (MCS)	Defines modulation and coding scheme via 3GPP lookup table
Transport Block Scaling	Configures scaling factor for transport block size
New Data Indicator	Indicates if resource allocation is for a re-transmission
Redundancy Version	Indicates the puncturing pattern after channel coding
HARQ Process Number	Indicates the HARQ process to use the resource allocation
Downlink Assignment Index	Updates the number of accumulated number of transmissions requiring acknowledgment for HARQ
TPC Command for Scheduled PUCCH	Used for closed loop power control as UE is directed to increase, decrease, or maintain power
PUCCH Resource Indicator	Instructs UE to use a specific PUCCH resource when returning HARQ acknowledgments
PDSCH to HARQ Feedback Timing	Determines number of slots between reception of the PDSCH and transmissions of the HARQ acknowledgments
Random-access Preamble Index	For PDCCH Order, specifies preamble for contention free random access or triggers contention based random access
UL/SUL Indicator	Indicates whether resource allocation is for the normal uplink carrier or the supplemental uplink carrier
SS/PBCH Index	Indicates SS/PBCH block for random access
PRACH Mask Index	Indicates PRACH occasion used for random access
System Information Indicator	Indicates whether PDSCH resource allocation for transmission of system information is for SIB1 or other
Reserved Bits	Added to ensure all variants of DCI 1_0 have equal size
Padding	Added to match size of DCI 1_0 and 0_0

Table 6. DCI 1_1 PDSCH Resource Allocations (Standard). Source: [9], [17].

Field	Description of Field Contents
Identifier for DCI Format	Differentiates DCI format 0_1 and 1_1
Carrier Indicator	Configures cross carrier scheduling
Bandwidth Part Indicator	Identifies BWP for frequency domain resource allocation
Frequency Domain Resource Assignment	Allocates a source of resource blocks in the frequency domain
Time Domain Resource Assignment	Determines slot offset, PUSCH Mapping Type, starting symbol and number of allocated symbols
VRB to PRB Mapping	Indicates if interleaving is used on PDSCH
PRB Bundling Size Indicator	Sets precoding for all contiguous Physical Resource Blocks within a Precoding Resource Block Group
Rate Matching Indicator	Sets rate matching to puncture PDSCH resources due to 'Reserved Resources'
Zero Power CSI Reference Signal Trigger	Triggers aperiodic Zero Power (ZP) CSI Reference Signal resources
Transport Block 1 MCS, NDI, RV	Modulation Coding Scheme, New Data Indicator, Redundancy Version for Transport Block 1
Transport Block 2 MCS, NDI, RV	Modulation Coding Scheme, New Data Indicator, Redundancy Version for Transport Block 2
HARQ Process Number	Indicates the HARQ process to use the resource allocation
Downlink Assignment Index	Updates the number of accumulated number of transmissions requiring acknowledgment for HARQ
TPC Command for Scheduled PUCCH	Used for closed loop power control as UE is directed to increase, decrease, or maintain power
PUCCH Resource Indicator	Instructs UE to use a specific PUCCH resource when returning HARQ acknowledgments
PDSCH to HARQ Feedback Timing	Determines number of slots between reception of the PDSCH and transmissions of the HARQ acknowledgments
Antenna Ports	Indicates which logical antenna ports the UE should use
Transmission Configuration Indication	Dynamically changes Quasi Co-Location assumptions for the PDSCH
SRS Request	Triggers SRS Resource Sets configured for aperiodic trigger
CBG Transmission Information	Used to configure Code Block Groups to transmit downlink data
CBG Flushing Out Information	Indicates if set of Code Block Groups being retransmitted can be combined with previous transmissions
DMRS Sequence Initialization	If transfer precoding disabled, initializes pseudorandom sequence which populates DMRS Resource Elements
Padding	Included if UE receives 1_1 in multiple search spaces

c. *DCI 2_0, 2_1, 2_2, and 2_3 UE Group Common Signaling*

The DCI format 2_0, which notifies a group of UEs of the slot format, is shown in Table 7, where each indicator represents a command to a different UE in the group. The DCI format 2_1, which notifies a group of UEs of a pre-emption for time critical transmission is shown in Table 8. While the recovery of a DCI format 2_1 message only reveals the group of UEs that are not conducting time critical transmissions, in a broader UE tracking environment, this information descrambled could be of some value. Table 9 shows the DCI format 2_2, which is used to transmit TPC commands for PUCCH and PUSCH, which could certainly be valuable in understanding that the connection has become stronger or weaker for that group requiring the TPC command. Finally, the DCI format 2_3, which is used to transmit TPC commands for SRS is shown in Table 10, and similarly may be indicative of a change in the strength of connection requiring an increase or decrease in power.

Table 7. DCI 2_0 Provision of Slot Format Indicators. Source: [9], [17].

Field	Description of Field Contents
Slot Format Indicator 1	Identifies slot format configuration for first UE
Slot Format Indicator 2	Identifies slot format configuration for second UE
…	…
Slot Format Indicator n	Identifies slot format configuration for n^{th} UE

Table 8. DCI 2_1 Provision of Pre-Emption Indications. Source: [9], [17].

Field	Description of Field Contents
Pre-Emption Indication 1	Specifies time/frequency resources pre-empted for first UE
Pre-Emption Indication 2	Specifies time/frequency resources pre-empted for second UE
…	…
Pre-Emption Indication n	Specifies time/frequency resources pre-empted for n^{th} UE

Table 9. DCI 2_2 Provision of Closed Loop Power Control Commands for PUCCH and PUSCH. Source: [9], [17].

Field	Description of Field Contents
Block Number 1	PUCCH and PUSCH TPC command for first UE
Block Number 2	PUCCH and PUSCH TPC command for second UE
…	
Block Number n	PUCCH and PUSCH TPC command for n^{th} UE
Padding	Depends upon size of DCI 1_0

Table 10. DCI 2_3 Provision of Closed Loop Power Control Commands to SRS. Source: [9], [17].

Field	Description of Field Contents
Block Number 1	SRS Request and TPC command for first UE
Block Number 2	SRS Request and TPC command for second UE
…	…
Block Number n	SRS Request and TPC command for n^{th} UE
Padding	Depends upon size of DCI 1_0

In this chapter, we have provided an overview of mobile communication metrics, the background behind 5G physical channels, a method to recover RNTIs in the blind in the PDCCH, and a detailed description of the RNTIs and DCI messages as directed by the 3GPP NR standards. In the next chapter, we present our methodology to optimize the recovery of RNTIs for different codeword lengths E, and payload lengths A, across different mobile environments and attacker goals.

THIS PAGE INTENTIONALLY LEFT BLANK

III. METHODOLOGY

In Chapter II, we provided an overview of mobile communication metrics, the 5G physical channels and the method developed in [3] to recover RNTIs through modified polar decoding and syndrome matching. In this chapter, we provide our methodology to optimize the recovery of RNTIs for different payload and codeword combinations and assess user activity based on the recovered RNTIs and DCI information. At the end of this chapter, we include a RNTI recovery walkthrough to demonstrate the optimization steps and how a user's activity can be assessed.

A. RECOVERY OF USER ACTIVITY

The methodology presented in this section, as shown in Figure 4, attempts to find the optimal parameters of modified polar decoding and syndrome matching to recover RNTIs and decode DCI messages without *a priori* knowledge of the RNTI. First, the assessed length of the codeword E, is used to determine the probable payload length, A, of the DCI message encoded. Next, the threshold values τ, to be used in the RNTI recovery method developed in [3] are evaluated. If the initial A and τ values attempted in the model do not successfully recover the RNTI, the payload and threshold search spaces are expanded. Once RNTIs are recovered, knowledge of $P_{success}$, which is the probability that a RNTI will be recovered for a given intercepted PDCCH message and P_{error}, which is the probability that a syndrome not generated from the correct RNTI is passed by τ, allow for optimization of the model for further RNTI recoveries. For the recovered RNTIs, the associated DCI messages can be descrambled and decoded to reveal different types of user activity, such as a change to geographical location or a change in reliability mode from normal to URLCC.

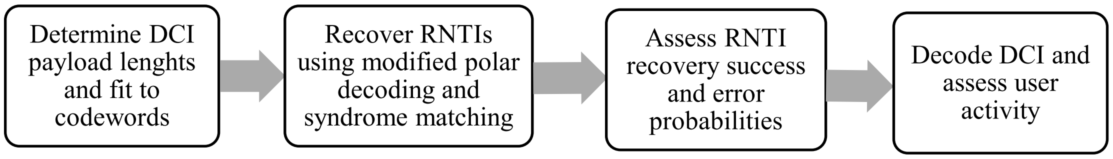

Determine DCI payload lenghts and fit to codewords
Recover RNTIs using modified polar decoding and syndrome matching
Assess RNTI recovery success and error probabilities
Decode DCI and assess user activity

Figure 4. Method to Optimize Recovery of UE Activity from PDCCH Messages without *A Priori* Knowledge of the RNTI

B. DCI PAYLOAD ANALYSIS

A DCI message, sent on the PDCCH, carries information directing the UE how to operate on the PUSCH and PDSCH or directs group signaling for SRS power, pre-emption, and slot-format. The allowed lengths for DCI messages are 12-140 bits [5]. To recover the RNTI in the blind, the actual length of the DCI payload of length A must be determined. To test all possible A lengths would require up to 128 different payload trials. Testing each payload can consume significant resources as the RNTI recovery algorithm attempts to match with up to 2^{15} possible scrambling sequences. This section considers methods to find the most probable DCI payload lengths in the PDCCH.

1. Expected DCI Payload Lengths

While DCI message size ranges from 12-140 bits, in practice DCI 0_0, 0_1, 1_0, and 1_1 formats will likely be around 70-80 bits in length, including the 24-bit CRC [7]. There will be up to four different DCI message sizes in total and a UE will in the blind attempt to decode each size until a message is recovered or all sizes are exhausted [9]. DCI 1_1 and DCI 0_0 formats will have unique sizes, DCI 1_1 format is matched in size with DCI 0_1 format, and the fourth size is utilized by messages activating features through DCI 2_0, 2_1, 2_2, and/or 2_3 formats [9], [22]. DCI 0_0, 0_1, 1_0, and 1_1 formats are typically scrambled by a C-RNTI initiated sequence while the DCI 2_0, 2_1, 2_2, and 2_3 formats are typically scrambled by a RNTI unique to the purpose of the respective messages; e.g., the TPC-SRS-RNTI scrambles DCI 2_3 format [9], [22]. Overall, the DCI 0_0, 0_1, 1_0, and 1_1 formats are the most desirable target as those DCIs contain the PDSCH and PUSCH control information directly related to user activity and are usually all associated with the same C-RNTI. The DCI 2_0, 2_1, 2_2, and 2_3 formats are typically associated with a RNTI shared by a group of UEs and without further information would not uniquely identify a specific UE [9]. For optimal RNTI recovery, an initial search space of DCI payload lengths of 70–80 bits is prudent. If searching the initial range is unsuccessful in recovering the RNTI, a pragmatic approach would be to iteratively expand the search out from the 70–80-bit range (e.g., 69, 81, 68, 82, etc.).

2. Maximum DCI Length Due to Probability of Block Error

For a given E, as A increases, the number of frozen bits F available for polar coding error correction decreases, reducing the code rate r and the error correction capability of the polar code. For mobile communications, a maximum $P_{bler} = 10^{-3}$ is typical in practice [20]. To determine the maximum payload to maintain the desired P_{bler} for a given E, the PDCCH encoding and decoding process are simulated in MATLAB to find the payload at which $P_{bler} = 10^{-3}$ as shown in Figure 5. Of note, the P_{bler} will be affected by the decoding method (LLR SC is used), the SNR, and the number of antennas used. Greater computational power of the decoder, higher SNR, and more antennas will allow for larger payloads to be sent within a codeword before exceeding the acceptable limit of $P_{bler} = 10^{-3}$ [20]. The probability of block error in a simulation can be calculated by the number of block errors per total number of blocks sent, where any bit error in a block results in a block error [20]. The expectation is that P_{bler} will increase as increased A results in a smaller F. Larger E will be able to support larger A as they have more bits available to be allocated as frozen bits.

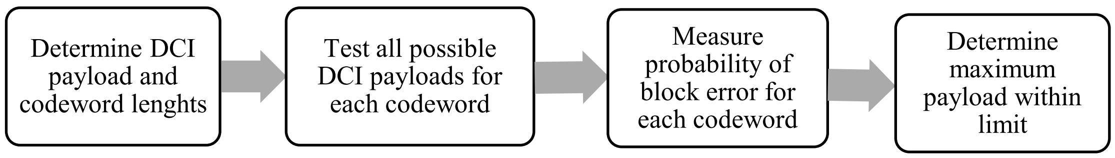

Determine DCI payload and codeword lenghts
Test all possible DCI payloads for each codeword
Measure probability of block error for each codeword
Determine maximum payload within limit

Figure 5. Methodology to Determine Maximum Payload Length, A for a Codeword Length, E within $P_{bler} = 10^{-3}$

C. RNTI RECOVERY METHODOLOGY

In [3], it was demonstrated that a RNTI can be recovered in the blind from intercepted PDCCH messages with a much-improved efficiency to a brute force approach. This section describes the method to optimize the method to recovery a RNTI shown in Figure 6. As described in detail in Section II.C, for an unknown RNTI, an intercepted PDCCH sequence is demodulated, rate recovered, sub-block deinterleaved, and mapped from LLR values to bits [3], [9]. At this point, for a known RNTI, the scrambling sequence would be applied to descramble the bits prior to polar decoding [3]. To apply modified polar decoding, polar decoding is applied to the bits without descrambling as the scrambling sequence is unknown [3]. The result of the modified polar decoding process produces a sequence of bits defined as an error pattern of length ε [3]. In addition to $P_{success}$ and P_{error}, we define our RNTI recovery efficiency η_{RNTI} as the ratio of successful RNTI recoveries to the total RNTI syndromes passed by τ.

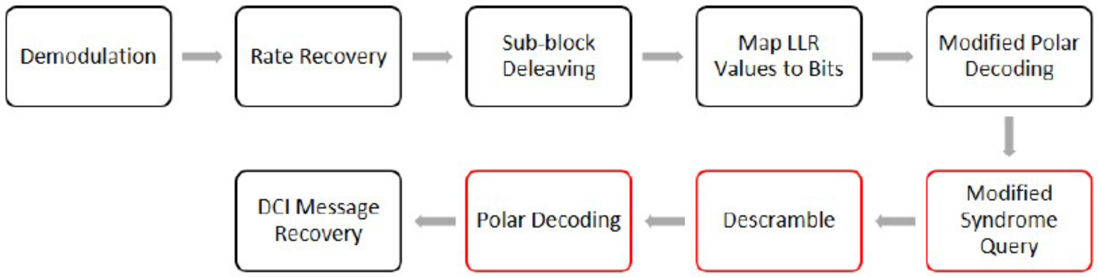


Figure 6. Modified Polar Decoding Method to Recover RNTIs in the Blind from Intercepted PDCCH Messages. Source: [3].

A given codeword E will contain $A+24$ bits of payload and CRC and F frozen bits as shown in Figure 7. As discussed in Section II.B.2, E is scrambled by a sequence initiated by the UE RNTI. Recall that there are 2^{16} possible RNTIs, but only the 15 least significant bits are used to initiate the scrambling sequence in the PDCCH encoding and decoding process [9]. We recall that

$$c_{init} = (n_{RNTI} \cdot 2^{16} + \eta_{ID}) \bmod 2^{31} \tag{3.1}$$

where c_{init} is the scrambling sequence initiator, n_{RNTI} is the RNTI assigned, and η_{ID} is the assigned cell ID [18]. As the inserted frozen bits are always zeros and the scrambling sequence when applied acts as non-random noise to uniquely flip these zeros, a syndrome table is pre-generated in which each 15-bit RNTI sequence is applied, and the resultant error syndromes are found [3]. The error pattern, $e_0^{\varepsilon-1}$ of length $\varepsilon = F$ recovered from the intercepted PDCCH sequence in the blind can then be uniquely matched to a syndrome $s_0^{\varepsilon-1}$ to identify the RNTI used in the intercepted PDCCH sequence [3]. Each payload and codeword combination will produce a unique set of syndromes, so if the payload length is unknown, the attacker must iteratively attempt the syndrome matching process for all probable payloads until a match is found [3].

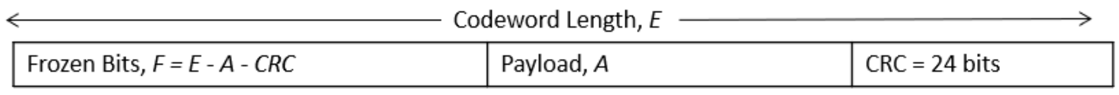


Figure 7. Relationship Between Codeword Length E, Payload A, Frozen Bits F, and CRC for DCI Messages Encoded in the PDCCH

1. Selection of Optimal Hamming Threshold

The optimization of τ_{HAM} is considered here and the optimization of τ_{LCS} is considered in section III.C.5 where the methods are compared. Recall from Section II.C.2 that the Hamming method considers the Hamming distance d_{HAM} between the two sequences and is set as a maximum threshold τ_{HAM} [3]. For RNTI recovery, the optimal τ_{HAM} is influenced by the mobile environment, specifically the number of UEs/RNTIs in use, the amount of PDCCH traffic intercepted, and the specific goals of the RNTI recovery activity. Consider two attackers who are intercepting PDCCH messages and recovering RNTIs. The first attacker intercepts frequent PDCCH messages with only a handful of UEs active on the gNB cell. As there are few UEs and many PDCCH messages, this attacker can set a low τ_{HAM} that results in a low individual $P_{success}$ but will still recover all RNTIs due to the large number of messages in which they can process. Once the RNTIs are recovered, this attacker can decode DCI messages for all the UEs in the mobile environment and potentially track their activity. A second attacker operates in an environment where they are infrequently intercepting PDCCH messages and have many UEs connected to the gNB. Since they receive messages infrequently, they need to maximize their chance of recovering an RNTI for each intercepted PDCCH message. This attacker will use their maximum resources on each intercepted message by setting a high τ_{HAM} to ensure they recover their targeted RNTI to track UE activity. This second attacker certainly operates in a more challenging environment, but instead of tracking all UEs, they may only be targeting one specific UE that can be confirmed to be linked to a recovered RNTI by other means.

To determine an optimal τ_{HAM}, consider a completely random error pattern of length ε. If we are to compare this random sequence to every possible sequence of length ε, the Hamming distance d_{HAM} between sequences will vary between $d_{HAM} = 0$ when the sequence is an exact match and $d_{HAM} = \varepsilon$ when the every bit is opposite. The median m_{HAM}, mean μ_{HAM} and variance σ_{HAM}^2 are determined empirically by calculating the Hamming distance between a randomly generated sequence and 2^{15} other randomly generated sequences, repeated for 100 trials. The results are shown in Figure 8 and Table 11, and we conclude that $m_{HAM} = \mu_{HAM} = \varepsilon / 2$.

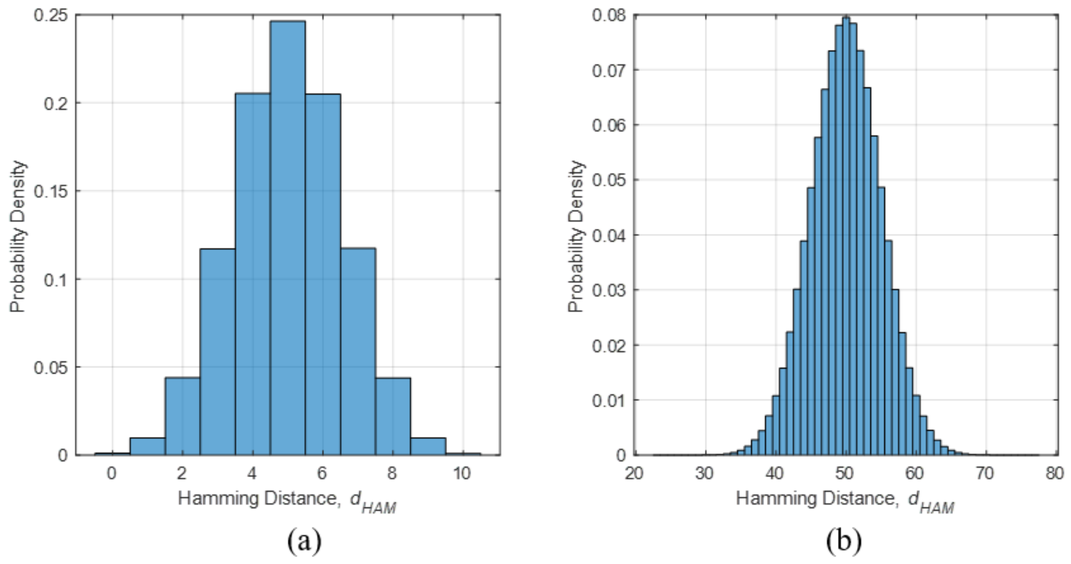


Figure 8. Normalized Histogram of Hamming Distance Results Between One Sequence and 2^{15} Randomly Generated Sequences of Length (a) $\varepsilon = 10$ and (b) $\varepsilon = 100$

Table 11. Statistics of Hamming Distance Between One Sequence and 2^{15} Randomly Generated Sequences of Length $\varepsilon = \{10, 100\}$

	$\varepsilon = 10$	$\varepsilon = 100$
Mean	5	50
Median	5	50
Variance	2.5	25

Further analyzing the relationship between ε and d_{HAM}, we find that the distribution of d_{HAM} follows a Gaussian distribution with mean $\mu_{HAM} = \varepsilon / 2$ and variance $\sigma^2_{HAM} = \varepsilon / 4$ where the relationships were found by analyzing $\varepsilon = [1:100]$ for 100 trials each, and the results are shown in Figure 9. Note in Figure 9 that the mean and median are identical lines; therefore, the probability density function for d_{HAM} can be represented by

$$f_{d_{HAM}}(d_{HAM}) = \frac{1}{\sqrt{\pi\varepsilon / 2}} e^{\frac{-2(d_{HAM} - \varepsilon/2)^2}{\varepsilon}} \tag{3.2}$$

where ε is the length of the error pattern, or more generally the length of a generic binary sequence. This distribution fits the data well for the trials of $\varepsilon = 10$ and $\varepsilon = 100$ as shown in Figure 10.

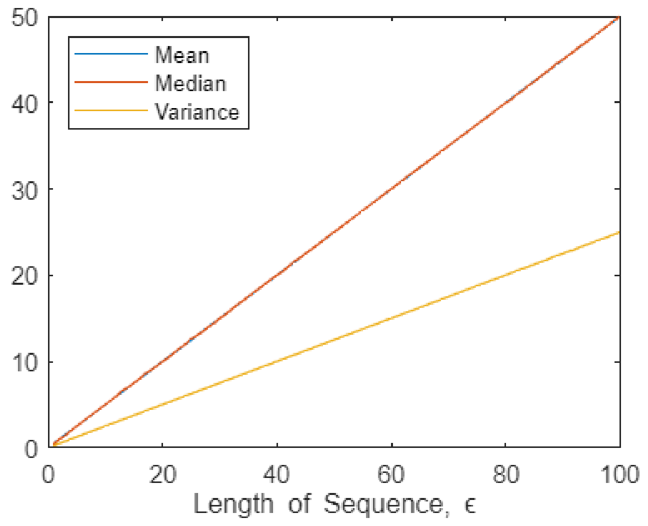


Figure 9. Mean, Median, and Variance for Hamming Results Between One Sequence and Randomly Generated Sequences of Length $\varepsilon = [1:100]$

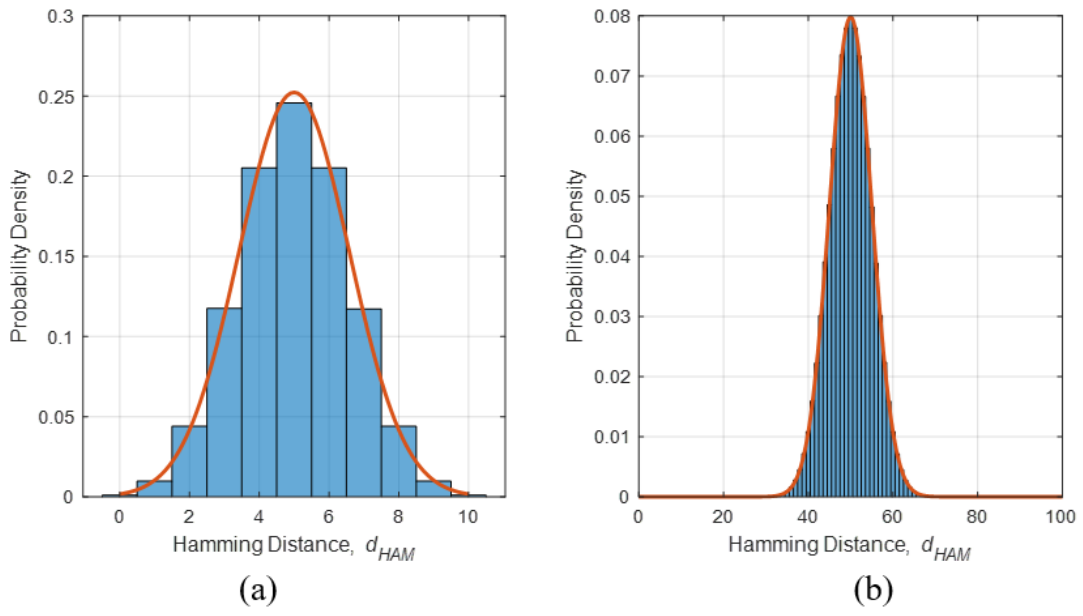


Figure 10. Normalized Results of Hamming Distance Results Between One Sequence and 2^{15} Randomly Generated Sequences of Length (a) $\varepsilon = 10$ and (b) $\varepsilon = 100$ Overlayed With Gaussian Probability Density Functions of $\mu_{HAM} = \varepsilon / 2$ and $\sigma^2_{ham} = \varepsilon / 4$

We can also consider the SNR and the channel bit error rate, P_b to determine an initial τ_{HAM}. Consider cases of $SNR = \{0, 5, 10\}$ dB for an error pattern of length $\varepsilon = 100$. In this case, we can calculate the probability of i channel bit errors as

$$P_{i,errors} = \binom{\varepsilon}{i} P^i_{b,QPSK} (1 - P_{b,QPSK})^{\varepsilon - i} \tag{3.3}$$

where $P_{b,QPSK}$ is the probability of QPSK channel bit error and ε is the length of the error pattern. We can estimate that

$$P_{b,QPSK} = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \approx Q\left(\sqrt{2SNR}\right) \tag{3.4}$$

We plot the results of (3.3) and (3.4) for $SNR = \{0, 5, 10\}$ dB and $\varepsilon = 100$ to generate a probability density function (PDF) as shown in Figure 11 and Table 12. From the probability density functions, the expected values of the number of channel bit errors for $SNR = \{0, 5, 10\}$ dB are $\{7.9, 0.60, 3.9\times10^{-4}\}$, respectively.

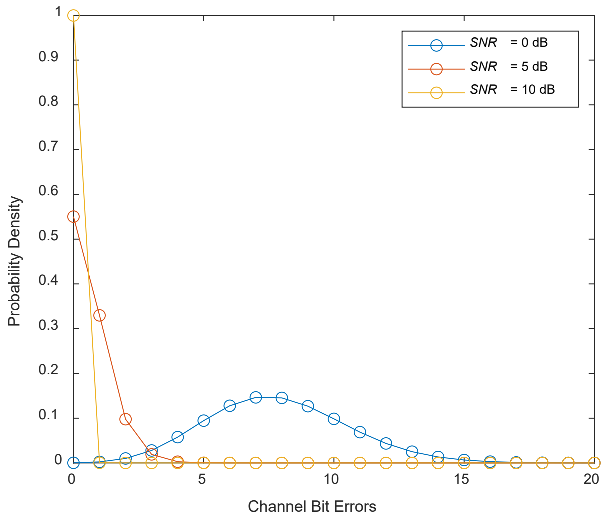


Figure 11. Probability Density of the Number of Channel Bit Errors in a Sequence of $\varepsilon = 100$ for QPSK and $SNR = \{0, 5, 10\}$ dB

Table 12. Probability Density for Number of Channel Bit Errors in QPSK

Number of Channel Bit Errors	Probability Density for $SNR = 0$ dB	Probability Density for $SNR = 5$ dB	Probability Density for $SNR = 10$ dB
0	2.77×10^{-4}	5.50×10^{-1}	1.00
1	2.36×10^{-3}	3.30×10^{-1}	3.87×10^{-4}
2	9.99×10^{-3}	9.77×10^{-2}	7.42×10^{-8}
3	2.79×10^{-2}	1.91×10^{-2}	9.38×10^{-12}
4	5.77×10^{-2}	2.78×10^{-3}	8.81×10^{-16}
5	9.45×10^{-2}	3.19×10^{-4}	6.55×10^{-20}
6	1.28×10^{-1}	3.03×10^{-5}	4.02×10^{-24}
7	1.46×10^{-1}	2.44×10^{-6}	2.09×10^{-28}
8	1.45×10^{-1}	1.70×10^{-7}	9.40×10^{-33}
9	1.27×10^{-1}	1.04×10^{-8}	3.72×10^{-37}
10	9.85×10^{-2}	5.66×10^{-10}	1.31×10^{-41}
11	6.88×10^{-2}	2.77×10^{-11}	4.15×10^{-46}
12	4.36×10^{-2}	1.23×10^{-12}	1.19×10^{-50}
13	2.52×10^{-2}	5.00×10^{-14}	3.13×10^{-55}
14	1.34×10^{-2}	1.86×10^{-15}	7.52×10^{-60}
15	6.54×10^{-3}	6.39×10^{-17}	1.67×10^{-64}
16	2.96×10^{-3}	2.03×10^{-18}	3.44×10^{-69}
17	1.25×10^{-3}	6.01×10^{-20}	6.57×10^{-74}
18	4.92×10^{-4}	1.66×10^{-21}	1.17×10^{-78}
19	1.81×10^{-4}	4.29×10^{-23}	1.96×10^{-83}
20	6.27×10^{-5}	1.04×10^{-24}	3.08×10^{-88}

If we can predict the number of errors in the channel for a given SNR, we can run an input sequence through modified polar decoding to predict the number of subsequent bit errors in the output sequence. Recall that modified polar decoding is performed by $\hat{n}_0^{N-1} = \hat{p}_0^{N-1} G_N$, where $\hat{p}_0^{N-1}$ are the polar coded bits estimated from the received LLR values, $\hat{n}_0^{N-1}$ are the received bits in which the polar coding process has been reversed, and G_N is the polarization matrix. Using a random sequence of $\varepsilon = 100$, we input the sequence to modified polar decoding and then repeated the modified polar decoding with bit errors added to the input sequence. The results of the modified polar decoded sequence bit errors, which is the equivalent of the Hamming distance d_{HAM} between the sequence with input errors and the initial sequence with no errors, are shown in Figure 12 and Table 13. We can then select a τ_{HAM} based on the SNR and subsequent $P_{b,QPSK}$. It will be found in

Chapter IV that the RNTI recovery process cannot achieve high $P_{success}$ when predicting modified polar decoded sequence bit errors given SNR as shown here. Instead, the number of polar decoded sequence bit errors should be used as a minimum value for τ_{HAM} as below this value, RNTI recoveries become very challenging.

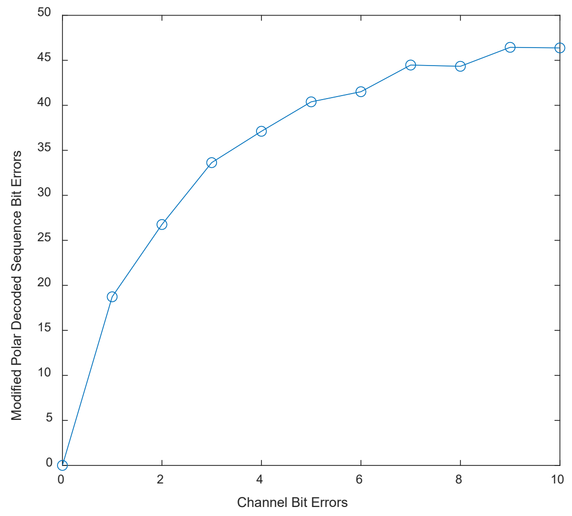


Figure 12. Output Hamming Distance, d_{HAM} in a Modified Polar Decoded Sequence versus Input Channel Bit Errors for $\varepsilon = 100$

Table 13. Modified Polar Decoding Output d_{HAM} Corresponding to Channel Bit Errors in Input for $\varepsilon = 100$

Number of Channel Bit Errors	Modified Polar Decoding Output d_{HAM}
0	0
1	17
2	24
3	30
4	33
5	37
6	39
7	41
8	42
9	44
10	44

Without knowledge of SNR, when considering a starting for RNTI recovery, using the median value $\tau_{HAM} = m_{HAM} = \varepsilon / 2$ would be expected to result in a $P_{error} \approx 0.50$ as half of the error patterns will be filtered through the threshold; however, P_{error} at $\tau_{HAM} = \varepsilon / 2$ is somewhat lower since in the RNTI recovery trials the RNTI iterative loop will break once the correct RNTI is found, which can be estimated to happen about halfway through the RNTI search space. As a result, a prudent estimate for $P_{error} \approx 0.50$ varies between $\tau_{HAM} = \varepsilon / 2$ and $\tau_{HAM} = 3\varepsilon / 4$. Once the baseline $P_{success}$ and P_{error} are determined, a target τ_{HAM} can be chosen, which would be optimized based on the number of PDCCH messages intercepted and the goals of the attacker.

For example, if ten PDCCH messages from the same RNTI are intercepted and an arbitrary Hamming distance threshold $\tau_{HAM} = 33$ results in $P_{success} = 0.30$, then by (3.1) where $P_{success(m)}$ is the overall probability of success for messages with the same RNTI and m is the number of messages processed,

$$P_{success(m)} = 1 - (1 - P_{success})^m \tag{3.5}$$

we conclude that we will have ultimately recovered the RNTI with $P_{success(10)} = 0.97$ and can decode all ten of the messages. It should be clear that an attacker attempting to optimize the computing resources expended should choose the strictest threshold that meets the desired RNTI recovery requirement.

2. Increased Payload Length

We can see from Figure 7, as the payload A increases, the number of frozen bits F decreases; therefore, the number of error pattern bits ε to be compared to a syndrome decreases as $\varepsilon = F$. In utilizing the Hamming method, which calculates the d_{HAM} between error patterns of length ε, there are now fewer bits in the calculation; therefore, d_{HAM} outputs will be lower and more patterns will fall within a fixed τ_{HAM}. In most cases, it is desirable to run the RNTI recovery algorithm at a consistent computational power; therefore, we want to maintain P_{error} constant across payload lengths. To achieve this goal, τ_{HAM} should be decreased to adjust for the decrease in ε caused by the increased A; however, the threshold is not the only factor affecting the results due to an increase in A. Once τ_{HAM} is adjusted to account for the change in ε, the increased A can affect $P_{success}$ independently of P_{error}. For one, F is smaller; therefore, there are less frozen bits for error correction purposes resulting in a lower code rate r that will make it more difficult to recover the RNTI as some errors will be uncorrectable. This has a particularly big impact in low SNR cases where correctly received bits are at a premium.

3. Increased Codeword Length

In practice, the codeword length E in which a payload is sent will vary based on the resources available on the PDCCH and the maximum P_{bler}; therefore, it is important to analyze the impact of E on RNTI recovery. Referring to Figure 7, for a given payload A, as E is increased, F increases significantly. In practice, increases in E would result in an increase from $E = 108$ to $E = 216$ and $E = 432$, which means an increase in F of 108 and 216 bits, respectively. As $\varepsilon = F$, the increase in F requires adjustments to τ_{HAM} as described in the previous section. Since more frozen bits are added, τ_{HAM} would need to be increased based on the number of frozen bits added to adjust for the impact of the Hamming calculation on $P_{success}$ and P_{error}. Once τ_{HAM} is adjusted, we consider the impact of increasing E on $P_{success}$ and P_{error}. As F is larger, r is higher, which improves $P_{success}$ as there is less of a likelihood that the RNTI will be unrecoverable. For this reason, $P_{success}$ and η_R are expected to be higher for larger E.

4. Impact of Signal-to-Noise Ratio

The SNR of the PDCCH signal received from the gNB is determined by

$$SNR = \frac{P_R}{P_N} = \frac{P_T G_T G_R}{P_N \left(\frac{4\pi d}{\lambda}\right)^2} \tag{3.8}$$

where P_R is the received signal power, P_T is the transmitted power, P_N is the noise power, G_T is the gain of the transmitter, G_R is the gain of the receiver, λ is the wavelength of the transmission, and d is the distance between the transmitter and receiver [12]. It is evident that SNR is influenced by the P_N (AWGN) and the distance of the UE from the gNB. If the UE experiences low SNR, the gNB will adjust P_T to ensure error-free communications. On the other hand, the intercepted messages from the gNB and the UE will likely be received at a different location, which could be further away from the gNB. Consequently, in this thesis we consider low SNR cases as a very real possibility for RNTI recovery operations.

The ability to recover a RNTI degrades significantly as SNR decreases due to the resultant increase in $P_{b,QPSK}$ as given by

$$P_{b,QPSK} = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \approx Q\left(\sqrt{2(SNR)}\right) \tag{3.9}$$

where E_b is the energy per bit, N_0 is the noise spectral density, and SNR is the signal-to-noise ratio [13]. To maintain $P_{success}$, τ_{HAM} must be increased to account for the increase in bit errors and this can be done to an extent at the expense of more computing resources expended. Further, if $P_{b,QPSK}$ is too high, there are cases where the error correction capabilities of the polar coding are exceeded resulting in uncorrectable errors and thus another case of an unrecoverable RNTI.

5. Hamming versus LCS Methods

The difference in performance between the Hamming and LCS Methods as analyzed in [3] for $A = 12$ found that the Hamming method had a slight advantage at the lower $SNR = 5$ dB while the LCS method had a slight advantage at the higher $SNR = 8$ dB when limiting $P_{error} \le 10^{-4}$. In this thesis, we evaluate the Hamming and LCS methods for $E = \{108, 128, 216, 256, 432, 512\}$ with $SNR = \{0, 5, 10\}$ dB and we consider much higher P_{error} cases that we expect an attacker to use. Further, we consider the optimization of RNTI recovery in which the Hamming and LCS methods have unique nuances. Recall that the LCS method considers the longest common substring d_{LCS} of matching bits between the two sequences, and if $d_{LCS} \ge \tau_{LCS}$, then the syndrome and error pattern are considered a match and the CRC is checked to determine if the RNTI is correct. The LCS distance d_{LCS} will vary between $d_{LCS} = 0$ where the sequences are the exact opposite of each other and $d_{LCS} = \varepsilon$ where the entire sequences match exactly. The mean μ_{LCS}, median m_{LCS}, and variance σ^2_{LCS} of d_{LCS} are determined empirically by calculating the LCS between a randomly generated RNTI and 2^{15} other randomly generated RNTIs, repeated for 100 trials. The results for $\varepsilon = 10$ and $\varepsilon = 100$ are shown in Figure 13.

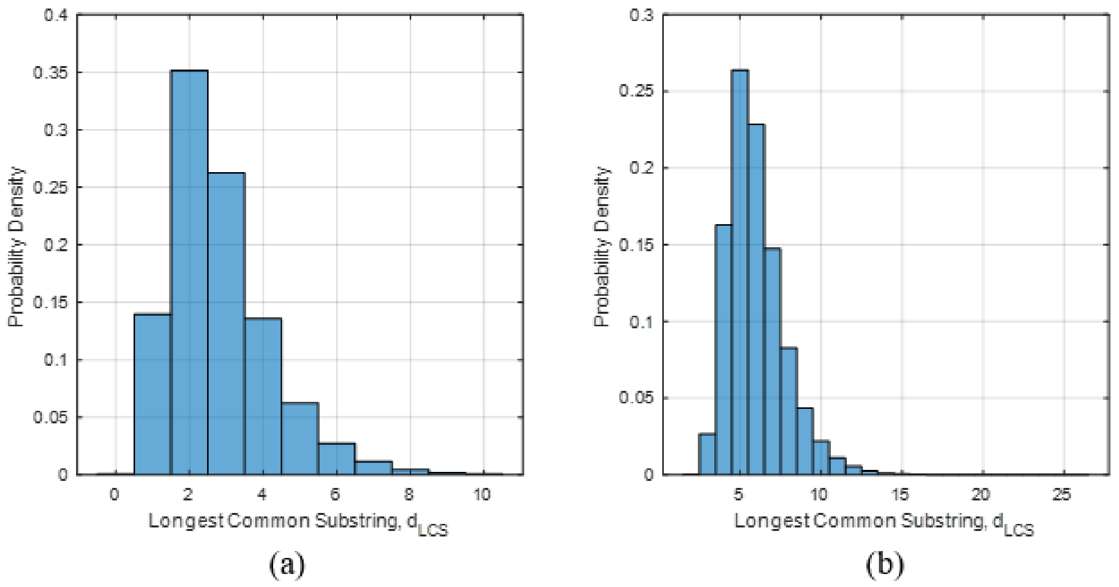


Figure 13. Histogram of Normalized LCS Results Between One Sequence and 2^{15} Randomly Generated Sequences of Length (a) $\varepsilon = 10$ and (b) $\varepsilon = 100$

We find that the relationship between d_{LCS} and ε much less straightforward in the LCS case than for the Hamming case, and so we run the same model for the range of $\varepsilon = [1:100]$ to attempt to empirically determine the relationship. We are interested in m_{LCS} because this value will allow us to establish a baseline τ_{LCS} for $P_{error} \approx 0.50$, which we can then adjust to be slightly less restrictive due to the fact that a RNTI is matched on average halfway through the set of 2^{15} possible values. However, we see in Figure 14 that the m_{LCS} does not change significantly over long changes in sequence length and then jumps at ε = {5, 10 22, 45, 90}. On the other hand, we observe that μ_{LCS} and σ^2_{LCS} increase logarithmically as ε increases, and specifically we found that $\mu_{LCS} \approx 3\log(\varepsilon)$ is a very close approximation as shown in Figure 14.

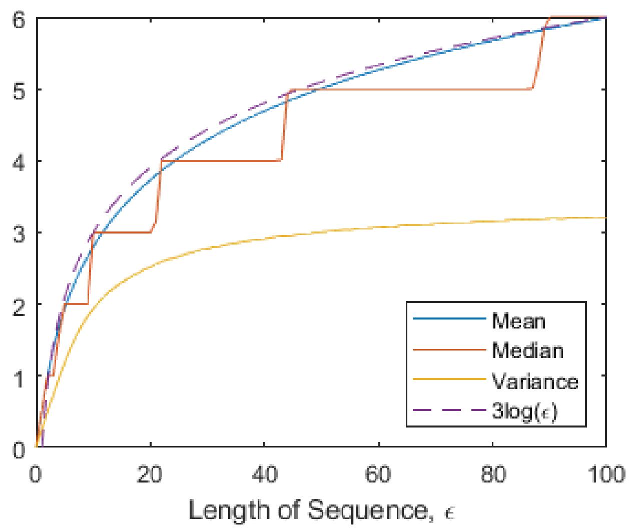


Figure 14. Mean, Median, and Variance for LCS Results Between One Sequence and 2^{15} Randomly Generated Sequences of Length $\varepsilon = [1:100]$

The statistics of LCS has been studied before for the purposes of molecular evolution and computational biology and [23] has established upper and lower bounds on the expected LCS while [24] has found the distribution can be scaled to match the Tracy-Widom distribution of the largest eigenvalue of a random matrix whose entries are drawn from a Gaussian unitary ensemble. Our results by inspection appear to also be similar to the aforementioned Tracy-Widom distribution, but it is important to note that the difference between our study and that of [23], [24] is that we are evaluating a LCS in which the indices of the sequence must also match while in [23], [24] the matching LCS can be in different locations in the two sets.

Consequently, when considering a starting τ_{LCS} for RNTI recovery, τ_{LCS} for $P_{error} \approx 0.50$ should be set according to Table 14 while extrapolating for $\varepsilon > 100$ utilizing $\mu_{LCS} = 3\log(\varepsilon)$ to guide the estimate of m_{LCS}. We also note τ_{LCS} is much less variable to changes in ε than τ_{HAM} with m_{LCS} being constant for long increases of ε, an important concept we will expand on in Chapter IV. In practice, the RNTI recovery method can be run at an assessed τ_{LCS}, and then the effect of E, A, and SNR can be estimated by measuring the resultant $P_{success}$, P_{error} and η_R. Once the RNTI recovery metrics are measured are measured, τ_{LCS} can be adjusted accordingly to achieve desired results.

Table 14. Median of LCS Between One Sequence and 2^{15} Randomly Generated Sequences of Length $\varepsilon = [1:100]$

	$\varepsilon \le 4$	$5 \le \varepsilon < 10$	$10 \le \varepsilon < 22$	$22 \le \varepsilon < 45$	$45 \le \varepsilon < 90$	$90 \le \varepsilon \le 100$
Median	≤ 2	2	3	4	5	6

As we did for the Hamming distance method, if the SNR is known or can be estimated, then we can find a minimum τ_{LCS} by analyzing the effect of channel bit errors, $P_{b,QPSK}$ on the modified polar decoded sequence LCS. Recall that Figure 11 and Table 12 show the probability density of the number of channel bit errors at $SNR = \{0, 5, 10\}$ dB. We apply those input channel bit errors to modified polar decoding and show the output of LCS results in Figure 15 and Table 15. We find through $P_{success}$ results in Chapter IV that the expected LCS in the modified polar decoded sequence should be used as a maximum τ_{LCS} for RNTI recovery as it does not result in high $P_{success}$ as expected, instead provides a baseline maximum.

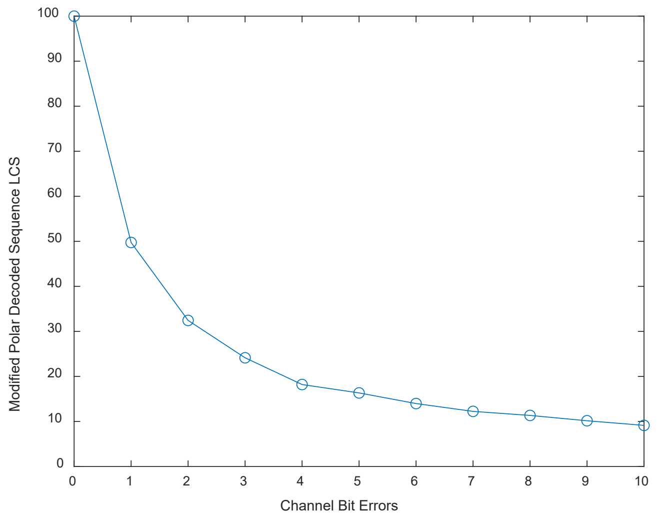


Figure 15. Output LCS, d_{LCS} in a Modified Polar Decoded Sequence versus Input Channel Bit Errors for $\varepsilon = 100$

Table 15. Modified Polar Decoding Output d_{LCS} Compared to Number of Input Channel Bit Errors for $\varepsilon = 100$

Number of Channel Bit Errors	Modified Polar Decoding Output d_{LCS}
0	100
1	46
2	32
3	24
4	19
5	16
6	14
7	12
8	22
9	10
10	9

D. ASSESSMENTS OF USER ACTIVITY

Once the RNTI recovery process has been optimized to meet the goals of the attacker, we now must consider what information the attacker has gained access to and how it can be used to track user activity. To do this, we analyze which RNTIs and which DCI parameters are exploitable.

1. Activity Recovered from RNTIs

RNTIs serve many purposes in the PDCCH, but first and foremost are used as an address for PDCCH messages to UEs [9]. The purpose of this section is to associate the recovery of certain RNTIs with specific user activity. When a UE first connects to a gNB through the PDCCH random-access procedure, the gNB assigns a RA-RNTI and then a TC-RNTI to allow for the addressing of the UE in the PDCCH [19]. Recovering a RA-RNTI or TC-RNTI addressed message is an indication that a new UE has entered the geographical area or that a UE is reconnecting from a disconnected state. Once the UE is authenticated and a C-RNTI is established, an increase in user activity could be associated with an increase in the frequency of PDCCH messages addressed to that C-RNTI. Further, it is hypothesized a change in the user's location resulting in a decrease in SNR could result in an increase in traffic addressed to that C-RNTI as the gNB may need to fine-tune UE control parameters more frequently.

The MCS-C-RNTI is used by the gNB to transition the UE to a higher or lower reliability modulation and coding scheme (MCS). This is done by changing the MCS index and/or MCS table itself for which the UE references when deciphering the fields of a DCI message [9]. If configured, the gNB will transition from scrambling the DCI messages to the UE with a C-RNTI to a MCS-C-RNTI [17]. This could be due to distance from the gNB, a degraded channel, or a change in operating mode [17]. The MCS-C-RNTI is frequently used for URLCC applications, but any UE could be configured with a MCS-C-RNTI [17]. Further, the TPC-PUSCH-RNTI and TPC-PUCCH-RNTI are used to adjust the power of the UE transmissions and can be utilized to increase power in an attempt to offset degraded channel conditions [17]. The use of a TPC-PUSCH-RNTI and TPC-PUCCH-RNTI again could indicated a change in user location or that the user is experiencing interference. Another RNTI that may be activated in degraded channel conditions is the SP-CSI RNTI. This RNTI is in use when a gNB may require more frequent CSI measurements from the UE such as when the UE is engaged in a voice or video call

[17]. Overall, these RNTIs associated with degraded channels can be linked to an observable change in geographic location to match the RNTI to a specific UE or be used to inform the attacker of a change in a tracked UE location.

Beyond tracking changes to the channel, other RNTIs are used to trigger a UE into certain modes of operation as signaled by the gNB. The MCS-C-RNTI was previously discussed and could fall into this category for high reliability applications, such as those used by URLCC devices. Additionally, the CS-RNTI is used by the gNB to trigger a configured scheduling mode in which the UE has a configured grant resource allocation in which it transmits rather than receiving a resource allocation on the PDCCH each time it requires an uplink transmission [17]. The CS-RNTI is used by the gNB to manage applications with a predictable and periodic traffic pattern, such as VoIP or URLCC communications and to conserve signaling resources in machine-type communications (MTC), which require small and infrequent packets [17]. In the first case, recovering a CS-RNTI can alert an attacker that a UE is making a VoIP call or using a similar application. In the latter cases, a CS-RNTI recovered can identify a device has transitioned to a URLCC mode or a mode in which only MTC are in use.

The INT-RNTI is used to address a group of UEs when the gNB requires an interruption for time-critical transmissions in the downlink [19]. This RNTI is sent to a group of UEs and while little information is revealed about the group, recovery of the INT-RNTI provides knowledge that a separate UE, not included in the group, is conducting time-critical transmissions. Lastly, one caveat of RNTIs used to scramble DCI 0_0, 0_1, 1_0, and 1_1 messages is that the RNTI recovery methodology presented in this thesis is not able to deduce which type of RNTI is scrambling the DCI message in the PDCCH without further information. For example, if only one RNTI is recovered, there is no way to know if it is the C-RNTI or a MCS-C-RNTI or a CS-RNTI as all scramble the same types of DCI messages. To gain valuable intelligence from these RNTIs, UE and RNTI relationships must be continuously tracked to identify new RNTIs and changes in traffic patterns of active RNTIs. Table 16 summarizes possible UE activity and RNTIS that could be indicative of that activity.

Table 16. Possible UE Activity Recovered from RNTIs Recovered in the PDCCH

UE Activity	RNTIs Indicative of Activity
UE has changed geographic location or is experiencing interference	RA-RNTI TC-RNTI C-RNTI MCS-C-RNTI TPC-PUSCH-RNTI TPC-PUCCH-RNTI TPC-SRS-RNTI SP-CSI-RNTI TC-RNTI RA-RNTI
UE is in a URLCC mode (critical IoT)	MCS-C-RNTI
UE is in a SPS mode (VoIP call, URLCC, MTC)	CS-RNTI SP-CSI-RNTI
A UE in the cell is conducting time-critical transmissions	INT-RNTI
A UE in the cell is first connecting or re-connecting to the cell	RA-RNTI, TC-RNTI

2. Activity Recovered from DCI Messages

Once the RNTI is recovered, the DCI message that is scrambled by the RNTI can be fully recovered, which reveals further information about the activity of the UE. DCI messages contain a myriad of information directing the UE how to operate on the PUSCH and PDSCH channels and what PDCCH features are activated for the UE. Similar to the RNTI, the information recovered from DCI messages can reveal user activity to include geographic location changes and mode of operation. The full list of information carried by DCI messages is listed in Section II.D.2 (specifically, see Tables 3–10).

First and foremost, changes to a UE geographical location can be revealed through information updated by the gNB in DCI messages. Recall that the received power is given by

$$P_R = \frac{P_T G_T G_R}{L_C} = E_b R_b \tag{3.10}$$

where the increase in an increase in L_C leads to a decrease in P_R and E_b, which by

$$P_{b,QPSK} = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{3.11}$$

causes an increase in $P_{b,QPSK}$ [12], [13]. This increase in $P_{b,QPSK}$ causes an increase in the information bit error rate P_b. The requirement to maintain P_b should trigger the gNB to change DCI fields to direct the UE to offset the higher P_b either by increasing P_T, changing the MCS or if necessary adjust other DCI fields to compensate, e.g., HARQ reporting and CSI reporting. Working backwards, if we observe a change in DCI fields directing a higher P_T or otherwise changing requirements to combat high P_b, we can deduce that the UE may be moving further away from the gNB. Further, potential changes to geographic location can be correlated with the timing advance (TA) assigned to the UE transmissions. A TA is typically assigned as a medium access control (MAC) layer control element (CE), which is a header applied to the MAC layer in the PBCH, PRACH, or PDSCH [19]. Once a RNTI is recovered and DCI messages decoded, the UE TA can be recovered by monitoring the PDSCH frequency and time resources assigned in the DCI 1_0 and 1_1 messages for the TA MAC CE.

A change in location resulting in a degraded channel would result in the gNB directing the UE to change to a more reliable MCS in the PUCCH and PDSCH. This is accomplished by changing the bit value of the MCS field [5]. The gNB could also adjust the HARQ Process Number field to change to a more frequent HARQ reporting mode [17]. A change in HARQ reporting could also be revealed in the Downlink Assignment Index, which determines the number of transmissions requiring acknowledgment before a HARQ request is sent or the PUCCH to HARQ Feedback Timing Indicator for which the gNB could increase the number of slots between PDSCH reception and the transmission of HARQ acknowledgment in a degraded channel [17].

Further, the TPC Command for Scheduled PUSCH and the TPC Command for Scheduled PUCCH fields are used as a gNB command to the UE to increase power on the PUSCH or PUCCH, respectively, in the event of a degraded channel [17]. The New Data Indicator field may be more frequently be set to zero in a degraded channel as this field indicates a re-transmission [17]. Similarly, the TB Block Scaling field can be adjusted by the gNB to increase redundancy [17]. Lastly, the SRS Resource Indicator, SRS Request, and CSI Request fields can also be adjusted by the gNB to direct the UE to send more SRS and/or CSI messages in a degraded channel situation [17]. While this discussion was initiated for a degraded channel due to a change in geographic location, the same conclusions could be drawn for a degraded channel caused by interference.

In the initial access case, the Random-access Preamble Index field would be used for the PDCCH Random-access Procedure, indicative that a UE that is first connecting or re-connecting to the gNB [17]. DCI messages also reveal indications of URLCC mode and carrier configuration. In the MCS field, frequently used with devices in URLCC mode, this could reveal a factory IoT device or other type of UE designed for URLCC [17]. The Uplink/Supplemental Uplink Indicator and Carrier Indicator fields, which reveal that a device is configured on multiple uplinks or carriers [17]. Table 17 summarizes possible UE activity and DCI fields that could be indicative of that activity.

Table 17. Activity Recovered from DCI Fields

UE Activity	DCI Fields Indicative of Activity
UE has changed geographic location or is experiencing interference	Modulation and Coding Scheme HARQ Process Number TPC Command for Scheduled PUSCH TPC Command for Scheduled PUCCH TPC Command for SRS SRS Resource Indicator SRS Request CSI Request New Data Indicator TB Scaling Indicator Downlink Assignment Index PUCCH to HARQ Feedback Timing Indicator Random-access Preamble Index
UE is in a URLCC mode	Modulation and Coding Scheme
UE is being configured across multiple carriers or links	Uplink/Supplemental Uplink Indicator Carrier Indicator
A UE in the cell is conducting time-critical transmissions	Pre-emption Indication

E. ACTIVITY RECOVERY DEMONSTRATION

In this section, we will demonstrate the recovery of user activity from a DCI message starting from the PDCCH interception. This example evaluates a specific payload-codeword combination to demonstrate the methodology to recover activity information through RNTI recovery and highlights the parameters that will later be adjusted to optimize the RNTI recovery process over a broad range of payloads, codewords, and channel conditions.

1. DCI Encoding of Control Information

In this example, let us assume the UE is moving away from the gNB; therefore, the gNB will reduce the complexity of the uplink MCS and increase the requested power from the UE to maintain a constant P_b. Note that on the gNB transmitter side, the transmitted power of the downlink will likely be adjusted as well, but since the gNB controls the transmitter, the gNB has no need to send out a message to the UE to do so. It is assumed that the gNB and UE are using the Fallback DCI formats and, therefore, the gNB will assign the updates to the UE in a DCI 0_0 PUSCH Resource Allocation (Fallback) message.

The UE operating on the PUSCH is assumed to initially be using MCS index table 4, index, $I_{MCS} = 19$, which corresponds to 64QAM, target code rate, $r = 0.554$, spectral efficiency $\eta_{spectral} = 3.3223$. As the UE has moved away from the gNB, we will assume the MCS is reduced to MCS index table 4 $I_{MCS} = 3$, which corresponds to QPSK, $r = 0.245$ and $\eta_{spectral} = 0.4902$ [17]. In the DCI 0_0, the MCS field is encoded at the bit level as [0 0 0 1 1]. Additionally, the gNB and UE are assumed to be using closed loop power control, and the gNB will issue a transmit power control (TPC) command to increase the UE transmitted power on the PUSCH. To accomplish this, the DCI 0_0 field TPC Command for Scheduled PUSCH field is set to [1 0], which selects the third element in the set of {-4, -1, 1, 4} and directs a power increase from the UE of 1 dB [17]. All other DCI fields are assumed to remain constant for this example. As laid out in Table 18, once the DCI parameters are selected, the DCI 0_0 message at the bit level is

$a_0^{39} = [0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]$, where a_0^{39} is the DCI payload bit string of length 40 bits.

Table 18. DCI 0_0 Encoding for a_1 to Trigger Decrease in MCS and Increase in TPC Power Control

Field	Number of Bits	Bits Assigned
Identifier for DCI Formats	1	0
Frequency Domain Resource Assignment	Depends on size of uplink BWP	0110011101101
Time Domain Resource Assignment	4	0010
Frequency Hopping Flag	1	0
Modulation and Coding Scheme	5	00011
New Data Indicator	1	0
Redundancy Version	2	11
HARQ Process Number	4	1000
TPC Command for Scheduled PUSCH	2	10
Padding	6 (depends on DCI 1_0 size)	000000
Uplink/Supplemental Uplink Indicator	1 or 0	1

2. PDCCH Encoding of DCI

The DCI is now encoded for transmission in the PDCCH. An RNTI is randomly assigned in the range of [1,65534], and an arbitrary cell ID $\eta_{ID} = 10$ is used. To prepare the DCI for transmission, the PDCCH encoding and modulation process as described in Section II.B.2 is performed. First, 24 CRC bits are calculated and inserted by scrambling and interleaving. The resultant block is encoded through polar coding followed by sub-block interleaving and rate matching to the codeword length E, then scrambling with an RNTI initialized sequence. Finally, the DCI message, now encoded in a codeword of length E is modulated and sent as QPSK symbols [9]. For this example, $E = 216$ was selected, which is a typical value for a PDCCH codeword [22]. During rate matching, the polar coded sequence, which expands the block to $N = 256$ bits, is shortened to $E = 216$ bits by removing frozen bits (puncturing). The scrambled $E = 216$ bits are QPSK modulated and transmitted as 108 complex QPSK symbols.

3. Recovery of RNTI Through Syndrome Matching

This section follows the methodology developed in [3] to recover the RNTI in the blind by using modified polar decoding and syndrome matching. To simulate real channel conditions, AWGN noise is added and then the received QPSK signals are demodulated to log-likelihood ratio (LLR) values. First, the $E = 216$ bit sequence is rate recovered to the polar coding block size of $N = 256$ by reversing the puncturing step of the encoding process. Next, the sub-block interleaving is reversed and the LLR values are mapped to bits. Modified polar decoding is then performed to recover an error pattern of length $\varepsilon = F$ where the error bit indices are uniquely affected by the RNTI initiated scrambling sequence. The error pattern is compared to a pre-generated syndrome table of all possible 2^{15} RNTI initiated scrambling sequences for the combination of $E = 216$ and $A = 40$. In this example, the Hamming distance thresholds used are τ_{HAM} = {5, 14, 24, 33, 43, 52, 62, 71, 81, 90}.

In each τ_{HAM} case, if the Hamming distance between the received error pattern and an entry in the syndrome table is less than τ_{HAM}, then that syndrome is considered a match and the associated RNTI is used to descramble the received PDCCH message and decode the DCI data bits and CRC bits. If a CRC calculated from the DCI data bits matches the CRC recovered from the PDCCH message, then the syndrome is considered correct and the RNTI is recovered. The τ_{HAM} range was chosen to evaluate $P_{success}$ and P_{error} over a broad range of cases from $\tau_{HAM} = 5$ where very few, if any, Hamming distance calculations outside of the correct RNTI sequence will meet the threshold and $\tau_{HAM} = 90$ where we approach the brute force case of evaluating every RNTI possibility. We know that $\varepsilon = F = E - A - 24 = 152$; therefore, we expect $P_{error} = 0.50$ to fall between $\tau_{HAM} = \varepsilon / 2 = 76$ and $\tau_{HAM} = 3\varepsilon / 4 = 114$. Once we select a τ_{HAM} to meet our RNTI recovery goals, the threshold can be held relatively constant as described in Section 3.E.4.

4. Evaluation of RNTI Recovery Success and Error Probabilities

We ran 300 trials at each τ_{HAM} to evaluate the impact of τ_{HAM} on $P_{success}$, P_{error} and η_R. The results, as seen in Figure 16, show the expected monotonic increase in $P_{success}$ and P_{error} with τ_{HAM}. The estimate of $P_{error} = 0.50$ between $\tau_{HAM} = \varepsilon / 2 = 76$ and $\tau_{HAM} = 3\varepsilon / 4 = 114$ is validated in Figure 16 (b). Further, we see in Figure 16 (b) that P_{error} increases significantly above $\tau_{HAM} = 60$ when we are still achieving $P_{success} \approx 0.50$ as seen in Figure 16 (a); therefore, if we were operating in an environment in which we are intercepting frequent PDCCH messages, this is a τ_{HAM} that uses resources very efficiently. However, if we are in an environment where we want to ensure we recover RNTIs as a very high $P_{success}$, then we need to consider Figure 17, which shows RNTI recovery efficiency η_R as we attempt to increase the number of RNTIs recovered. The η_R relationship between $P_{success}$ and P_{error} is mostly linear, but we keep in mind that this is a log-log scale, so while η_R is linear, the resources expended to achieve a higher $P_{success}$ are increasing exponentially.

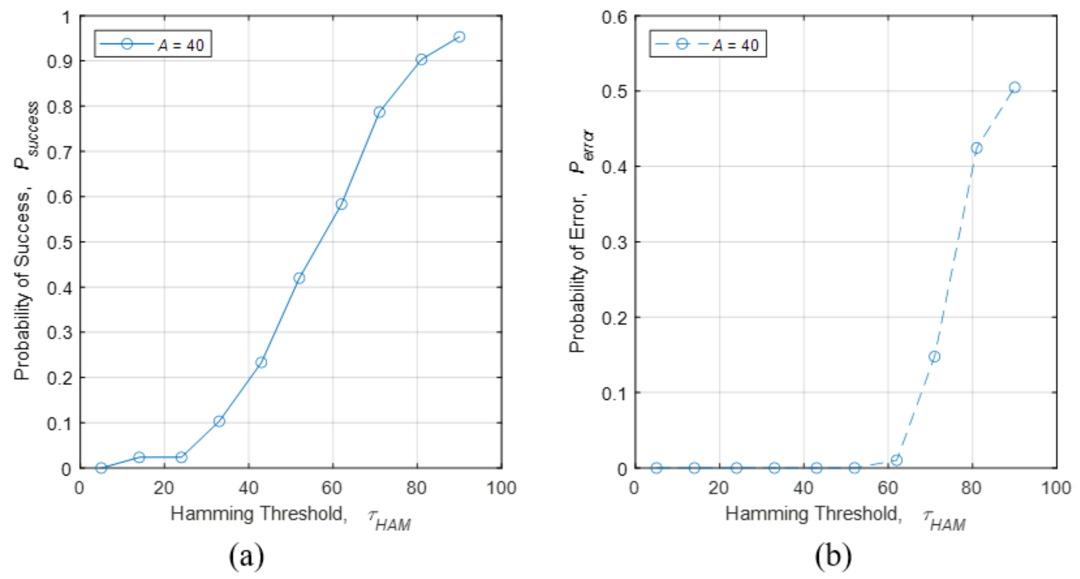


Figure 16. Hamming Method RNTI Recovery (a) $P_{success}$ and (b) P_{error} versus τ_{HAM} for $A = 40$, $SNR = 5$ dB, $E = 216$, and DCI Payload a_0^{39}

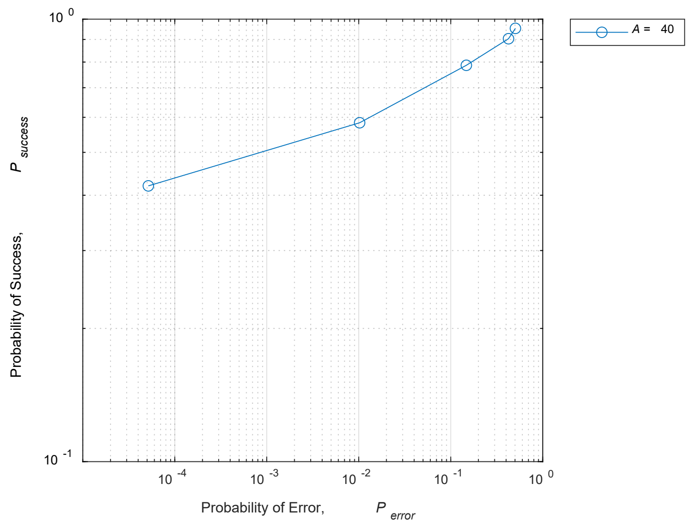


Figure 17. Hamming Method RNTI Recovery Efficiency for $A = 40$, $E = 216$, $SNR = 5$ dB and DCI Payload a_0^{39} as shown by $P_{success}$ versus P_{error}

5. Decoding of DCI Information and Assessment of UE Activity

As the RNTI is recovered, the attacker can now decode the DCI message and recover the payload bits that carry DCI field information. In this case, we recover the DCI bits

$$\hat{a}_1 = [0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]$$

where $\hat{a}_1$ represents the DCI payload bits recovered. In the fields relevant for UE activity tracking, first the MCS is field is recovered as [0 0 0 1 1], which corresponds to MCS index table 4 $I_{MCS} = 3$ directing QPSK, $r = 0.245$, and $\eta_{spectral} = 0.4902$, a decrease in MCS from our initial value. Second, the TPC Command for Scheduled PUSCH field is revealed to be [1 0], which directs a power increase from the UE of 1 dB.

Note that in this example, we assume that the attacker has some prior knowledge of the UE state, presumably from previous RNTI recovered DCI messages. Since the attacker knows that the MCS field has been changed to a lower code rate and spectral efficiency and the TPC Command for Scheduled PUSCH has been increased, the attacker can deduce that the UE is moving further from the gNB. With this information and a possible method to associate the UE to this activity (e.g., location of interference, visual of UE moving, only UE in area, etc.), the RNTI can be linked to a UE for further tracking. It is not beyond noting that if the attacker were to generate interference on a target UE to intentionally degrade the channel, they could use this to determine which RNTIs DCI messages are directed to offset the degraded channel and now can associate that RNTI with the UE being interfered with.

In this chapter, we have provided the methodology for optimizing the recovery of RNTIs in different mobile environments. We discussed the impact on RNTI recovery of A, E, SNR, and Hamming versus LCS methods. We presented an empirical statistical analysis of the Hamming and LCS distances to understand how τ can be used as a lever to control the RNTI recovery $P_{success}$, P_{error} and η_R. Finally, we provided a RNTI recovery walkthrough to demonstrate how an attacker could adjust parameters and methods to optimally recover user activity in a specific scenario.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SIMULATION RESULTS AND ANALYSIS

In Chapter III, we presented the methodology to determine probable DCI payload lengths and to optimize recovering RNTIs using modified polar decoding and syndrome matching to decode DCI messages and assess user activity. This chapter provides the structure of the MATLAB models used for the DCI payload and RNTI recovery simulations and then presents the results for $E = \{108, 128, 216, 256, 432, 512\}$. For each E, results of RNTI recovery are presented for a range of payloads using the Hamming method. Further we consider RNTI recovery results at varying SNR and using the LCS method. Finally, we present a discussion of what UE activity can be recovered using the RNTIs recovered by these methods.

A. MAXIMUM DCI PAYLOAD MODEL

A $P_{bler} \le 10^{-3}$ is typical in mobile communications, so to find the maximum supported DCI payloads within this P_{bler}, a simulation was created utilizing the MATLAB 5G toolbox to model the encoding and decoding of the PDCCH channel as shown in Figure 18. First, randomly generated DCI payload of lengths $A =$ [12,140] are encoded onto a codeword of length E, QPSK modulated, and simulated to be sent over an AWGN channel. The recovered codeword is demodulated, decoded, and error corrected through the polar coding built in to the PDCCH encoding and decoding process. The recovered payload is then compared to the sent payload. If they are not identical, then a block error has occurred. The simulation is repeated for the codeword lengths $E = \{108, 128, 216, 256, 432, 512\}$. The output metric of this model is number of block errors per blocks sent or P_{bler} where it should be noted that any bit error in a block will result in a block error. We simulated 50,000 trials at $SNR = \{0, 5, 10\}$ dB and measured the number of block errors. The large number of trials was necessary as there are few errors when A is small or E is large as the polar coding applied in the PDCCH process has considerable error correction capabilities. The output data was analyzed to determine the maximum A for each E that meets the $P_{bler} \le 10^{-3}$ requirement.

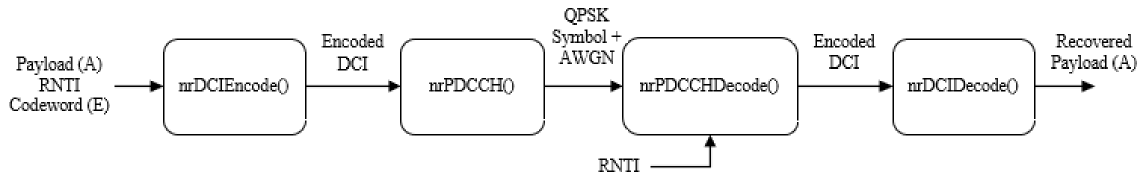


Figure 18. MATLAB Model to Determine Maximum Supported A for Each E for $P_{bler} \leq 10^{-3}$ Requirement

It is important to note that the polar decoding function in this model is accomplished by successive cancellation (SC) decoding, which is equivalent to successive cancellation list (SCL) decoding with list length, $L = 1$ [20]. In practice, the L may be higher, a typical value can be as high as $L = 8$, and this will allow for larger payloads to be sent within the codeword limits [20]. However, in this thesis, the SCL decoding of $L = 1$ sets a baseline and the assumed maximum payload can be adjusted as an input to the activity recovery model. Similarly, this thesis assumes one antenna, but additional antennas can improve the overall recovery of data bits [3].

B. MAXIMUM DCI PAYLOAD RESULTS

The results of 50,000 trials at all possible payloads, $A =$ [12:140] , codeword lengths $E =$ {108, 128, 216, 256, 432, 512}, and varying $SNR =$ {0, 5, 10} dB are presented in Figures 19, 20, and 21, respectively. The results confirm that larger codewords can support larger payloads within $P_{bler} \leq 10^{-3}$. In fact, for $SNR = 5$ dB the codewords $E =$ {432, 512} are not limited by the maximum $P_{bler} = 10^{-3}$ and can support up to the maximum DCI payload of 140 bits. For the $SNR = 5$ dB case, which will be evaluated in detail, the maximum DCI payloads are listed in Table 19.

Under poor channel conditions, as shown in Figure 19 for $SNR = 0$ dB, the maximum DCI payloads are further limited. In this case, codewords $E =$ {108, 128} cannot support any DCI payloads within the desired P_{bler}, codewords $E =$ {216, 256} can only support small payloads of approximately $A \leq 50$, and codewords $E =$ {432, 512} now are limited and cannot support the maximum $A = 140$. If the SNR were this low in practice, the gNB would most likely increase $E > 512$ or take other means outside of channel coding to manage the high P_{bler}. On the other hand, under good channel conditions, as shown in Figure 21 for $SNR = 10$ dB, the maximum DCI payload lengths meeting $P_{bler} \leq 10^{-3}$ are much higher. Codewords $E = \{216, 256, 432, 512\}$ can support all DCI payloads up to the maximum 140 bits while codewords $E = \{108, 128\}$ are limited by the P_{bler} but can support much higher payloads than the baseline $SNR = 5$ dB case.

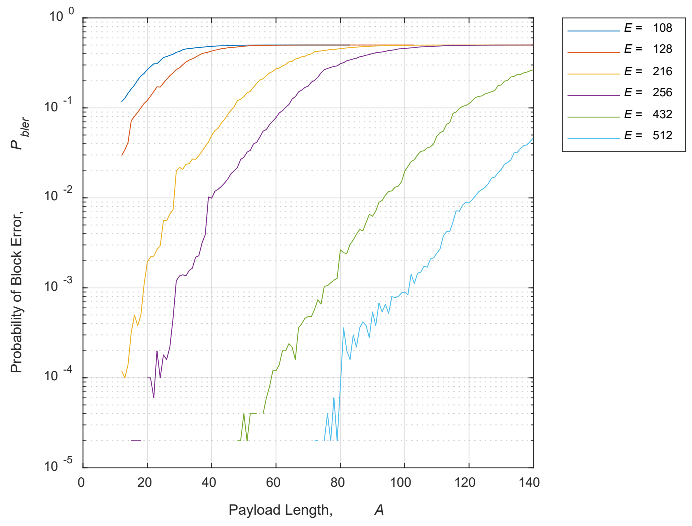


Figure 19. Probability of Block Error at Payload Lengths A = [12:140] and Codeword Lengths $E = \{108, 128, 216, 256, 432, 512\}$ with $SNR = 0$ dB

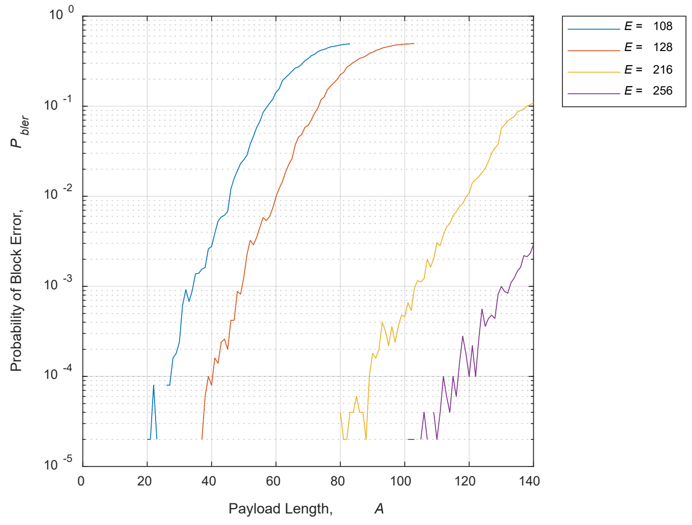


Figure 20. Probability of Block Error at Varied Payload Lengths $A = [12:140]$ and Codeword Lengths $E = \{108, 126, 216, 256\}$ with $SNR = 5$ dB

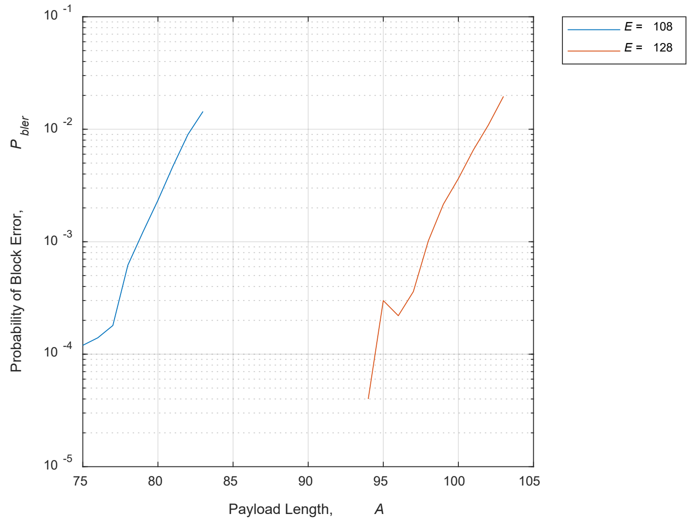


Figure 21. Probability of Block Error at Varied Payload Lengths $A = [12:140]$ and Codeword Lengths $E = \{108, 128\}$ with $SNR = 10$ dB

Table 19. Maximum Payload for $P_{bler} \leq 10^{-3}$ and $SNR = 5$ dB

Codeword Length	Maximum Payload
108	33
128	49
216	103
256	133
432	140+
512	140+

C. RNTI RECOVERY MODEL

The simulation to recover RNTIs from intercepted PDCCH messages is modeled using the MATLAB 5G toolbox and MATLAB code developed in [3]. The goal of the simulation is to input the bit stream of a PDCCH codeword without knowledge of the RNTI and ultimately recover the DCI bits from which user activity can be determined. While a brute force method can be attempted, the modified polar coding and syndrome matching technique presented in [3] is expanded to the full spectrum of payload lengths within several practical codeword lengths. The simulation metrics and thresholds are analyzed for changes to A, E, SNR, τ, and recovery method to determine optimal RNTI recovery for different mobile traffic environments and distinctive UE tracking goals.

1. Input Variables and Factors

First, the DCI payload range A and codeword lengths E are considered, and in this thesis we choose the A and E inputs to the RNTI recovery simulation to reflect values used in practice as much as possible. The codeword lengths in practice are $E = \{108, 216, 432\}$ [10]. As RNTI recovery becomes increasingly effective at large codeword lengths due to the increase in frozen and error pattern bits for evaluation, this simulation was limited in scope to codeword length up to $E = 432$ for practical values. For the practical values, simulations were also run at the associated polar coding codeword lengths $E = \{128, 256, 512\}$, where the codeword bits are matched to the polar coding bits, but due to the mechanisms of polar coding are constrained to $N = 2^n$ in size. The payloads A are spaced to explore the entire range of possible results for each E. The minimum allowed

DCI payload length is $A = 12$, and the maximum was chosen to be the lesser of the maximum allowed DCI payload $A = 140$ or the maximum A found for $P_{bler} \leq 10^{-3}$ as discussed in Section IV.B and presented in Table 19 [9]. As shown in Table 20, the A values to be tested were linearly spaced between the minimum and maximum A to produce five total values for $E = \{108, 128\}$ and ten total values for $E = \{216, 256, 432, 512\}$; more values were added in the latter cases to account for the larger range of possible payloads.

In Sections IV.D.1 through IV.D.4, the Hamming distance method is used to determine if an error pattern and a syndrome are considered a match; therefore, Hamming distance thresholds are set as simulation inputs. The maximum τ_{HAM} was chosen to be $\tau_{HAM} = E/2$ as this threshold value approaches the brute force case. For the minimum, $\tau_{HAM} = 5$ was chosen as this was shown by trial and error to significantly limit P_{error} while still recovering RNTIs. Recall that $E = \varepsilon + A + 24$ and a threshold between $\tau_{HAM} = \varepsilon/2$ and $\tau_{HAM} = 3\varepsilon/4$ will result in $P_{error} \approx 0.50$. Within the minimum and maximum τ_{HAM}, threshold values were linearly spaced to produce five total values for codewords $E = \{108, 128\}$ as shown in Table 20. Also shown in Table 20, for the larger codewords $E = \{216, 256, 432, 512\}$, ten threshold values were linearly spaced between minimum and maximum to produce ten total values, this time for the larger range of possible Hamming distances due to an increase in ε.

Table 20. MATLAB RNTI Recovery Simulation Inputs

Codeword Length (E)	Payload Lengths (A)	Hamming Threshold Values (τ_{HAM})
108	12, 17 23, 28, 33	5, 13, 21, 28, 36
128	12, 21, 31, 40, 49	5, 15, 26, 36, 46
216	12, 22, 32, 42, 52, 63, 73, 83, 93, 103	5, 14, 24, 33, 43, 52, 62, 71, 81, 90
256	12, 25, 39, 52, 66, 79, 93, 106, 120, 133	5, 17, 28, 40, 52, 63, 75, 87, 98, 110
432	12, 26, 40, 55, 69, 83, 97, 112, 126, 140	5, 26, 48, 69, 90, 112, 133, 154, 176, 197
512	12, 26, 40, 55, 69, 83, 97, 112, 126, 140	5, 31, 57, 83, 109, 134, 160 186, 212, 238

2. MATLAB Model for RNTI and DCI Recovery

The intercepted PDCCH message is first generated using the MATLAB 5G toolbox functions of `nrDCIEncode()` and `nrPDCCH()`, which are derived from 3GPP 5G standards TS 38.212 [9] and TS 38.211 [18], respectively. The function `nrDCIEncode()` takes the payload of length A, the RNTI, and the codeword length E as inputs and performs the CRC attachment, polar coding, and rate matching to output the encoded DCI bits. The input payload bits of length A and the RNTI are randomly generated in our model. Inputs to the function `nrPDCCH()` are the encoded DCI bits, η_{ID}, and RNTI, and outputs are the complex QPSK modulation symbols. The overall model used in MATLAB to generate the PDCCH QPSK symbols is shown in Figure 22. Next, AWGN is added to the QPSK symbols using `awgn()` to simulate realistic channel conditions.

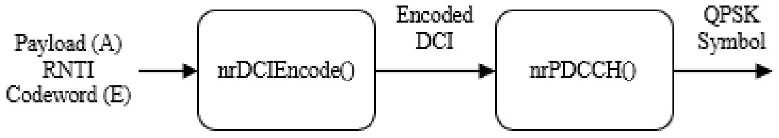


Figure 22. MATLAB Model for PDCCH Encoding Following 3GPP Standards [9] and [18].

At this point, if the RNTI is known, the functions `nrPDCCHDecode()` and `nrDCIDecode()` can be used to demodulate and decode the DCI message. However, since the PDCCH message is recovered in the blind, the modified polar decoding and RNTI recovery process demonstrated in [3] is used. First, the `nrSymbolDemodulate()` function is used, which demodulates the received QPSK + AWGN symbols using soft decision decoding into LLR values. Next, the steps of rate matching and sub-block interleaving are reversed from the encoding process and the LLR values are mapped to bits. The step that is omitted is the descrambling since the RNTI is unknown, thus the output block is $N = 2^n$ polar coded bits scrambled by a Gold sequence initiated by the RNTI. The modified frozen bits of length F of this scrambled sequence are extracted as the error pattern and compared via Hamming distance to the error pattern syndromes pre-generated for all possible RNTIs [3]. A schematic diagram of this method is shown in Figure 23, and the output of this stage is a syndrome of length $\varepsilon = F$ that matches the error pattern within a Hamming distance threshold τ_{HAM}. This is an iterative process that attempts to match each syndrome for all RNTI possibilities sequentially from $n_{RNTI} = 1$ through $n_{RNTI} = 2^{15}$ and breaks the loop only if a decoded CRC correctly matches the CRC calculated for the recovered DCI bits as described in the next few steps [3].

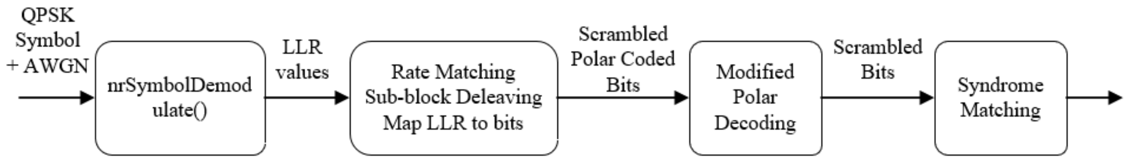


Figure 23. MATLAB Model for Modified Polar Decoding and Syndrome Matching to Recover RNTIs in the PDCCH as Developed in [3].

For an error pattern that matches a syndrome within the Hamming distance threshold, we then go back to the associated LLR values and perform the descrambling, rate matching and sub-block deinterleaving to recover the unscrambled polar coded bits [3]. These unscrambled bits are then decoded using `nrPolarDecode()` and recovered by `nrCRCDecode()` as shown in Figure 24. For the actual RNTI used to encode, `nrPDCCHDecode()` and `nrDCIDecode()` could replace the descrambling, rate matching, sub-block deinterleaving and polar decoding steps performed by manual functions `nrPolarDecode()` and `nrCRCDecode()`. However, since we do not know if the matching syndrome is associated with the correct RNTI, the process is broken down to allow for the calculation of the CRC from the DCI bits, which can be compared to the decoded CRC recovered from descrambling. To check the CRC, `nrCRCDecode()` is used and if the calculated CRC matches the decoded CRC, then the RNTI is determined to be correct and the iterative loop comparing error patterns to syndrome matches is ended, and the RNTI is considered recovered. If the calculated CRC does not match the recovered CRC, then the iterative process continues for $n_{RNTI} = 1$ through $n_{RNTI} = 2^{15}$. The RNTI from the next match within τ_{HAM} is used to descramble that syndrome match, and the process is repeated until the correct RNTI is found or the entire set of RNTI syndrome matches is evaluated [3].

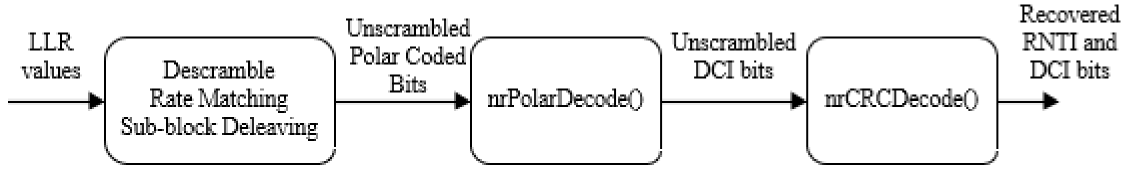

LLR values
Descramble Rate Matching Sub-block Deleaving
Unscrambled Polar Coded Bits
nrPolarDecode()
Unscrambled DCI bits
nrCRCDecode()
Recovered RNTI and DCI bits

Figure 24. MATLAB Model for RNTI Recovery for Matched Syndromes in the PDCCH as Developed from [3].

3. Simulation Metrics

As the received error pattern is compared to each RNTI associated syndrome, there are five possible outcomes as first presented in [3] and shown in Figure 25. To optimize the RNTI recovery process, the True Positive case should be maximized while minimizing the False Positives. This will optimize the processing required to recover the RNTI as each case in which a syndrome match is processed within the threshold, the descrambling, decoding, and CRC check calculations must be completed.

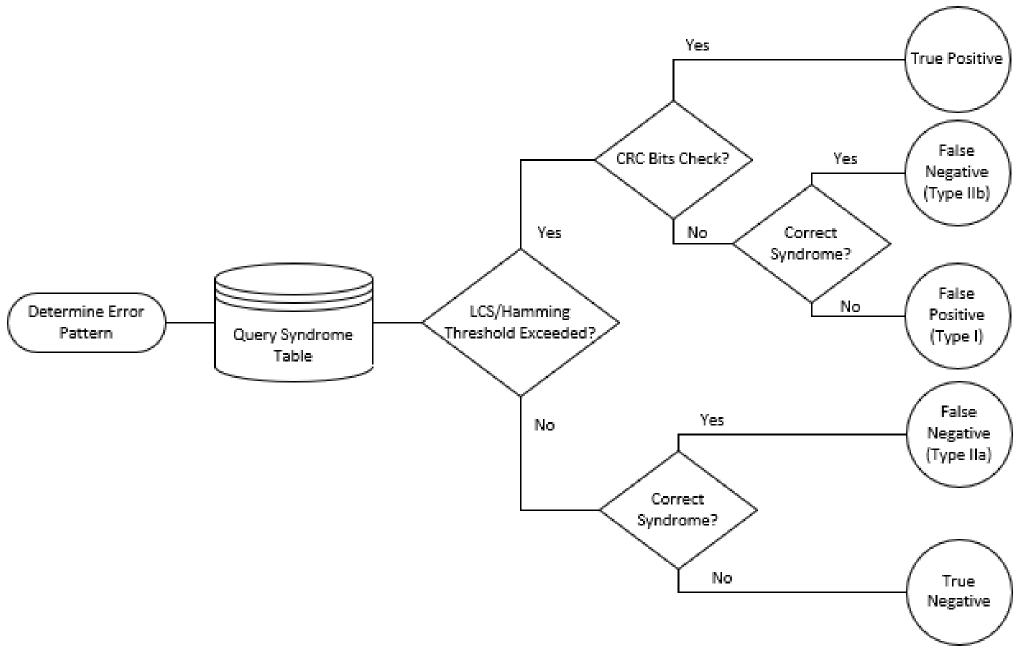


Figure 25. Decision Tree and Output Metrics for MATLAB RNTI Recovery Simulation. Source: [3].

Probabilities $P_{success}$ and P_{error} are the focus in this thesis as we are evaluating successfully recovered RNTIs ($P_{success}$) and the resources expended by processing syndromes within the threshold but not resulting in a recovered RNTI (P_{error}). To calculate our statistics in terms of simulation outcomes, we define the probability that a RNTI will be recovered for a given intercepted PDCCH message

$$P_{success} = \frac{n_{TruePositive}}{N_{trials}} \tag{4.1}$$

where $n_{TruePositive}$ is the number of True Positive outcomes, and N_{trials} is the number of trials or the number of PDCCH messages processed. A True Positive is the case where the correct RNTI syndrome is passed by τ, decoded, and verified as the correct RNTI by CRC check. Further, we define the probability that a syndrome not generated from the correct RNTI is passed by τ as

$$P_{error} = \frac{n_{TypeI} + n_{TypeIIb}}{2^{15} \cdot N_{trials}} \tag{4.2}$$

where n_{TypeI} is the number of False Positive (Type I) errors, $n_{TypeIIb}$ is the number of False Negative (Type IIb) errors, and N_{trials} is the number of trials or the number of PDCCH messages processed. A False Positive (Type I) error occurs when a syndrome is within τ but is not the correct RNTI syndrome, thus is rejected once the CRC bits are checked. A False Negative (Type IIb) will occur if the correct RNTI syndrome is within τ, but the CRC bits do not check due to bit errors, and in our model this will be counted within P_{error} since we don't know that the CRC check failure was in fact the correct RNTI. This case is only common in low SNR scenarios and since this RNTI syndrome is only one of 2^{15}, it does not statistically impact P_{error}. A False Negative (Type IIa) is a failure of the RNTI syndrome to be matched to the error pattern within τ and is accounted for as approximately $1-P_{success}$, where the approximate characterization is due to the possibility of a False Negative (Type IIb) error. Similarly, a True Negative is the case where the incorrect RNTI syndrome is not properly matched within τ and is accounted for as $1-P_{error}$.

The last statistic we define is the RNTI recovery efficiency η_R as the ratio of successful RNTI recoveries to the total RNTI syndromes passed by the τ, which we can calculate as

$$\eta_R = \frac{n_{TruePositive}}{n_{TypeI} + n_{TypeIIb} + n_{TruePositive}} = \frac{P_{success}}{P_{error} \cdot (2^{15}-1) + P_{success}} \approx \left(\frac{1}{2^{15}}\right)\frac{P_{success}}{P_{error}} \tag{4.3}$$

We note that for a brute force case, $\eta_R = \frac{P_{success}}{P_{error} \cdot 2^{15} + P_{success}} = \frac{1}{1\cdot(2^{15}-1)+1} = 0$, and for a perfect recovery case $\eta_R = \frac{P_{success}}{P_{error} \cdot 2^{15} + P_{success}} = \frac{1}{0\cdot(2^{15}-1)+1} = 1$. We caution the reader that $\eta_R = 1$ does not mean that all RNTIs are recovered but that any RNTI that is recovered is done with perfect efficiency, i.e., with no more extra computations than necessary.

Finally, we expand on the metrics established in [6] to explain how adjusting parameters can drive model outcomes, the factors that significantly influence the respective outcomes are described in Table 21. It is important to note that to achieve a higher probability of True Positive, directly increasing $P_{success}$, the threshold τ for matching error patterns to syndromes should be relaxed. However, this also increases the probability of False Positives, increasing P_{error}. Therefore, in a real-world mobile environment, it is recommended to establish $P_{success}$ and P_{error} for a τ resulting in $P_{error} \approx 0.50$, and subsequently adjust τ in real time to optimize RNTI recovery. As will be shown in Section IV.D.4, higher SNR results in an increase in True Positives without significantly effecting False Positives, thus resulting in a much higher recovery efficiency, η_R.

Table 21. RNTI Recovery Output Metrics. Source: [3].

Model Outcome	Description of Result	Significant Influence(s)
True Positive	Received frozen bit pattern matches syndrome within threshold, CRC bits check, the RNTI is successfully recovered	Relaxed threshold → higher likelihood Increased SNR → higher likelihood
False Negative (Type IIb)	For the correct RNTI, received frozen bit pattern matches syndrome within threshold, CRC bits do not check	Increased SNR → lower likelihood
False Positive (Type I)	Received frozen bit pattern matches syndrome within threshold, CRC bits do not check	Relaxed threshold → higher likelihood
False Negative (Type IIa)	For the correct RNTI, received frozen bit pattern does not match syndrome within threshold	Relaxed threshold → lower likelihood Increased SNR → lower likelihood
True Negative	Received frozen bit pattern does not match syndrome within threshold	Relaxed threshold → lower likelihood

D. RNTI RECOVERY RESULTS

The results presented in this section cover 300 trials for each payload length A and codeword length E combination evaluated at the Hamming and LCS threshold values τ

presented in Table 20. Sections IV.D.1 through IV.D.3 evaluate the effect of increased τ_{HAM}, increased A, and increased E on $P_{success}$, P_{error}, and η_R, respectively. In Sections IV.D.4 and IV.D.5, a codeword of $E = 216$ is used to evaluate the effect of high and low SNR and the differences between the Hamming and LCS methods, respectively.

1. Effect of Increased Threshold

This section considers the impact of increasing τ_{HAM} between an error pattern and a syndrome on $P_{success}$, P_{error}, and η_R for $A =$ {12, 22, 32, 42, 52, 63, 73, 83, 93, 103} with $E = 216$ across the range of τ_{HAM} = {5, 14, 24, 33, 43, 52, 62, 71, 81, 90}. The symmetry of the results across A, as shown in Figure 26, show that for larger A, τ_{HAM} can be decreased to maintain P_{error} constant, and this agrees with our analysis in Section II.C.2. We estimate from the results that, to maintain P_{error} constant, every increase in A of approximately 10 bits, which subsequently decreases F and ε by 10 bits, requires an decrease in τ_{HAM} of approximately five bits. This is consistent with $\mu_{HAM} = \varepsilon / 2$ where an increase to ε of 10 results in an increase to μ_{HAM} of five, which was derived in Section III.C.2. We will also find that this shift is SNR agnostic as we adjust SNR in Section IV.D.4. Typically, A will not be exactly known for the intercepted PDCCH message, thus P_{error} will be significantly impacted by the initial τ_{HAM}.

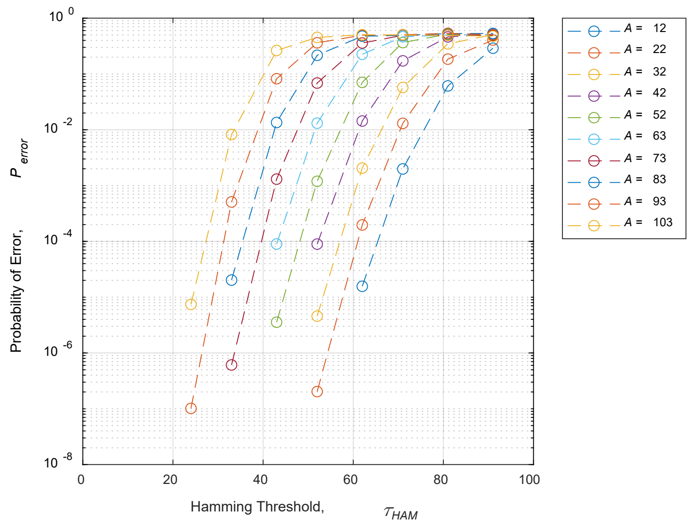


Figure 26. Hamming Method RNTI Recovery P_{error} versus τ_{HAM} for $E = 216$, $SNR = 5$ dB, and $A =$ {12, 22, 32, 42, 52, 63, 73, 83, 93, 103}

The significance of adjusting τ_{HAM} is shown Figure 27, where we observe that the $P_{success}$ is much more tightly grouped than P_{error} as we increase τ_{HAM}. We conclude that if τ_{HAM} is not properly adjusted, P_{error} can increase significantly with a less substantial increase in $P_{success}$, resulting in wasted resources in processing incorrect RNTIs. Another thing to note from Figure 27 is that for larger A and subsequent smaller ε, $P_{success}$ does increase faster as τ_{HAM} increases, and this is due to the fact that there less bits in ε, thus $\sigma^2_{HAM} = \varepsilon / 4$ is smaller. A prudent Hamming distance simulation must be adjusted regularly for probable DCI payload lengths; however, we will find in Section IV.D.5 that the LCS method does not share this property and is more robust to changes in A.

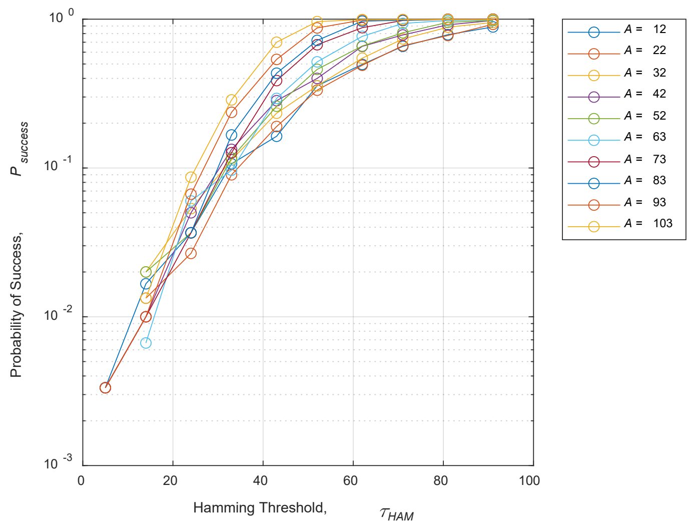


Figure 27. Hamming Method RNTI Recovery $P_{success}$ versus τ_{HAM} for $E = 216$, $SNR = 5$ dB, and $A =$ {12, 22, 32, 42, 52, 63, 73, 83, 93, 103}

2. Effect of Increased Payload Length

This section presents the individual results of increased A on $P_{success}$ and P_{error} for fixed codeword lengths $E =$ {108, 128, 216, 256, 432, 512}, SNR = 5 dB, and the Hamming distance method. Within a given E, A is expected to vary for different DCI messages, and in this section we analyze how the change in payload length impacts RNTI recovery.

a. Codeword Length of 108

The codeword length $E = 108$ is expected to be used in practice but will likely only support small payloads as the maximum DCI payload for $P_{bler} \le 10^{-3}$ at $SNR = 5$ dB is $A = 33$ as found in Section IV.B. For $E = 108$, the payloads of $A =$ {12, 17, 23, 28, 33} were evaluated at τ_{HAM} = {5, 13, 21, 28, 36}. Figure 28 (a) shows that $P_{success}$ increases monotonically and nearly linearly as τ_{HAM} increases across the tested range. $P_{success}$ is higher for larger A for a given τ_{HAM}, but this is only due to the higher P_{error} at these τ_{HAM} cases as more RNTIs are being filtered through the threshold. The P_{error} results in Figure 28 (b) show as expected that the largest A results in the highest P_{error} for a given τ_{HAM} due to the decrease in ε as explained in Section III.C.2. If we compare $P_{success}$ and P_{error} to evaluate the RNTI recovery efficiency η_R, as shown in Figure 29, we find that in fact smaller A can recover RNTIs more efficiently for a fixed P_{error}, and this is due to the increase in ε and F resulting in a higher code rate r as described in Section III.C.2.

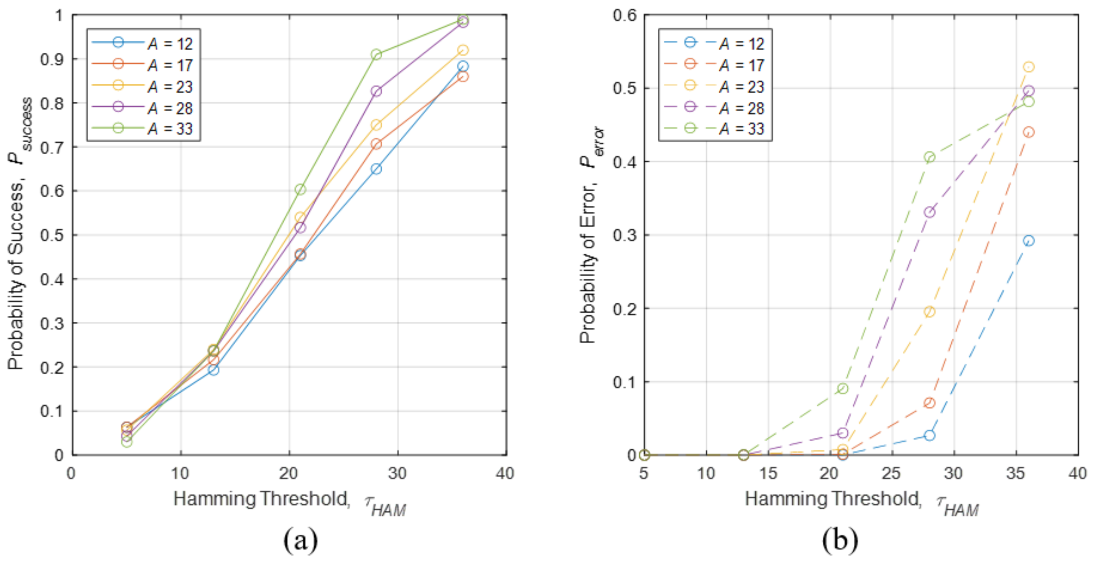


Figure 28. Hamming Method RNTI Recovery (a) $P_{success}$ and (b) P_{error} versus τ_{HAM} for $E = 108$, $SNR = 5$ dB, and $A =$ {12, 17, 23, 28, 33}

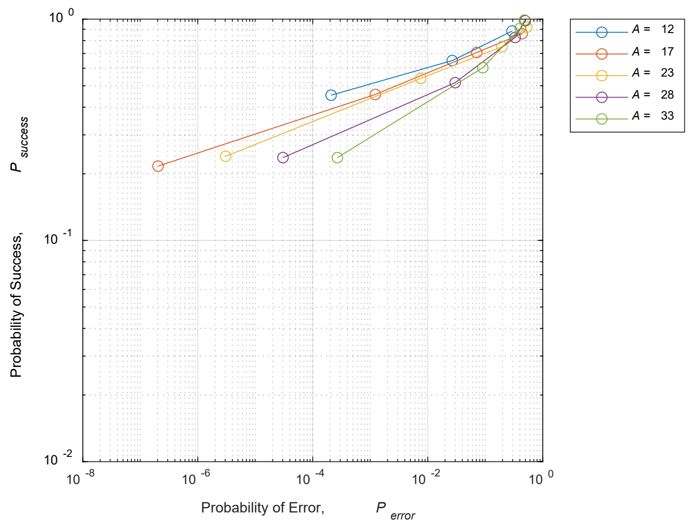


Figure 29. Hamming Method RNTI Recovery Efficiency for $E = 108$, $SNR = 5$ dB, and $A =$ {12, 17, 23, 28, 33} as shown by $P_{success}$ versus P_{error}

b. *Codeword Length of 128*

The codeword length $E = 128$ is a theoretical case to match the polar block coding length of $N = 128$ and would only support small payloads as the maximum DCI payload for $P_{bler} \leq 10^{-3}$ at $SNR = 5$ dB is $A = 49$ as found in Section IV.B. For $E = 128$, the payloads of $A =$ {12, 21, 31, 40, 49} were evaluated at τ_{HAM} = {5, 15, 26, 36, 46}. From Figure 30, we see that $P_{success}$ and P_{error} follow the same monotonically increasing trends as for $E = 108$, but as the payloads are spaced out more so are the results. If we compare $P_{success}$ and P_{error} to evaluate η_R, as shown in Figure 31, we again find that smaller A can recover RNTIs more efficiently for a fixed P_{error} due to the increase in ε and F resulting in a higher r as described in Section III.C.2.

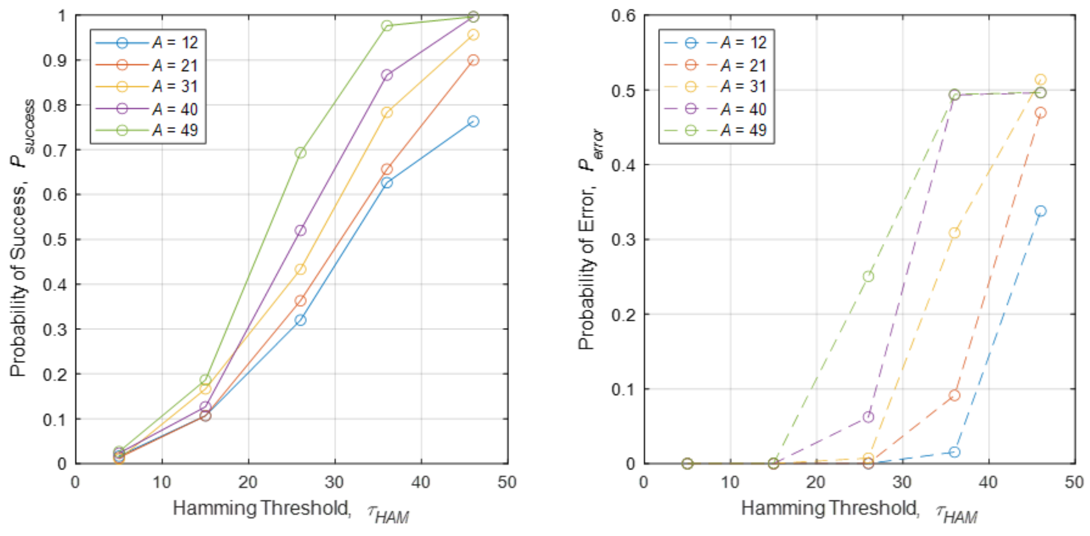


Figure 30. Hamming Method RNTI Recovery (a) $P_{success}$ and (b) P_{error} versus τ_{HAM} for $E = 128$, $SNR = 5$ dB, and Payloads $A =$ {12, 21, 31, 40, 49}

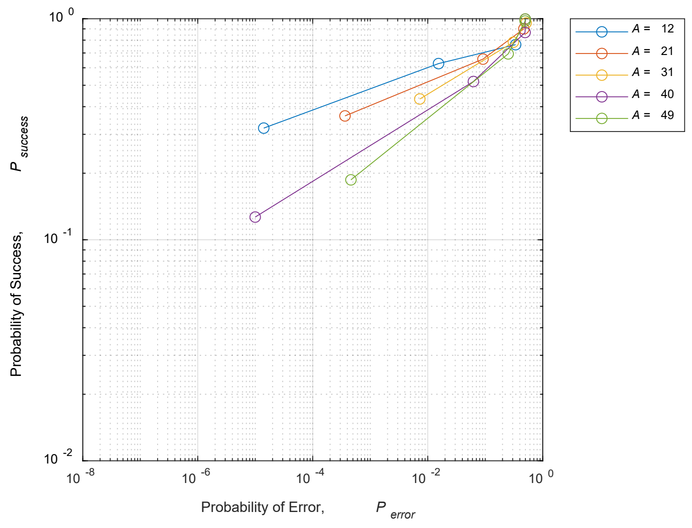


Figure 31. Hamming Method RNTI Recovery Efficiency for $E = 128$, $SNR = 5$ dB, and $A = \{12, 21, 31, 40, 49\}$ as shown by $P_{success}$ versus P_{error}

c. Codeword Length of 216

The codeword length $E = 216$ is expected to be used in practice and support most payloads as the maximum DCI payload for $P_{bler} \leq 10^{-3}$ at $SNR = 5$ dB is $A = 103$ as found in Section IV.B. For $E = 216$, the payloads of $A =$ {12, 22, 32, 42, 52, 63, 73, 83, 93, 103} were evaluated at $\tau_{HAM} =$ {5, 14, 24, 33, 43, 52, 62, 71, 81, 91}. From Figure 32, we see that $P_{success}$ and P_{error} follow the same monotonically increasing trends as for smaller codewords, but here for the larger A values we begin to see $P_{success}$ and P_{error} reach their maximums. $P_{success}$ maximizes and becomes flat as we approach $P_{success} = 1$ while P_{error} oscillates around the value of $P_{error} = 0.50$. This oscillation can be explained by the fact that the RNTI recovery simulation will find the correct RNTI on average halfway through the RNTI search space, thus will hover around $P_{error} = 0.50$ for a brute force case, which is what we experience for the large A and high τ_{HAM} values chosen for this simulation.

The brute force case is not unique to the E length but depends on the length of the error pattern ε and τ_{HAM} value chosen. If we analyze $A = 103$ on $E = 216$, we find that $\varepsilon = E - A - 24 = 89$, and this will correspond to $\mu_{HAM} = \varepsilon / 2 = 44.5$ and $\sigma_{HAM} = \sqrt{\sigma_{HAM}^2} = \sqrt{\varepsilon / 4} = 4.72$ where σ_{HAM} is the standard deviation of the Hamming distance. From Figure 32 (b), we observe that $A = 103$ begins to oscillate around $P_{error} = 0.50$ at $\tau_{HAM} = 62$, which is 3.7 standard deviations from μ_{HAM}, which due to the Gaussian distribution results in over 99.9% of the RNTI syndrome patterns within $\tau_{HAM} = 62$. Finally, we compare $P_{success}$ and P_{error} to evaluate η_R, as shown in Figure 33, we continue to demonstrate that smaller A can recover RNTIs more efficiently for a fixed P_{error} due to the increase in ε and F resulting in a higher code rate as described in Section III.C.2.

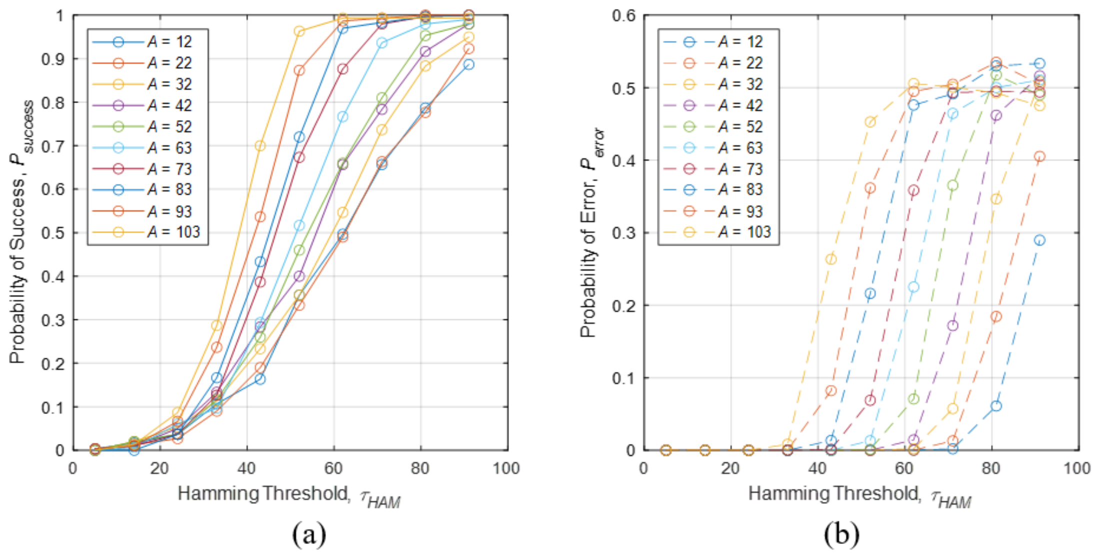


Figure 32. Hamming Method RNTI Recovery (a) $P_{success}$ and (b) P_{error} versus τ_{HAM} for $E = 216$, $SNR = 5$ dB, and $A =$ {12, 22, 32, 42, 52, 63, 73, 83, 93, 103}

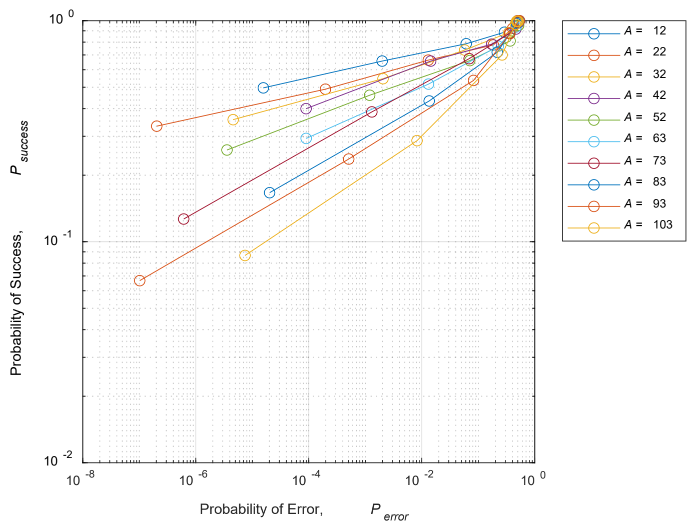


Figure 33. Hamming Method RNTI Recovery Efficiency for $E = 216$, $SNR = 5$ dB and $A =$ {12, 22, 32, 42, 52, 63, 73, 83, 93, 103} as shown by $P_{success}$ versus P_{error}

d. Codeword Length of 256

The codeword length $E = 256$ is a theoretical case to match the polar coding block length of $N = 2^8 = 256$ and would support almost all payloads as the maximum DCI payload for $P_{bler} \le 10^{-3}$ at $SNR = 5$ dB is $A = 133$ as found in Section IV.B. For $E = 256$, the payloads of $A =$ {12, 25, 39, 52, 66, 79, 93, 106, 120, 133} were evaluated at τ_{HAM} = $\{5, 17, 28, 40, 52, 63, 75, 87, 98, 110\}$. From Figure 34, we see that $P_{success}$ and P_{error} follow the same monotonically increasing trends as for smaller codewords, and we see $P_{success}$ and P_{error} reach their maximums. As we compare $P_{success}$ and P_{error} to evaluate η_R, as shown in Figure 35, we continue to demonstrate that smaller A can recover RNTIs more efficiently for a fixed P_{error}. For $E = 256$, an interesting case occurs at $A = 120$ as observed in Figures 34 and 35 where $P_{success} = 0$ for τ_{HAM} = {5, 17, 28, 40}, well below the expected $P_{success}$ for that payload.

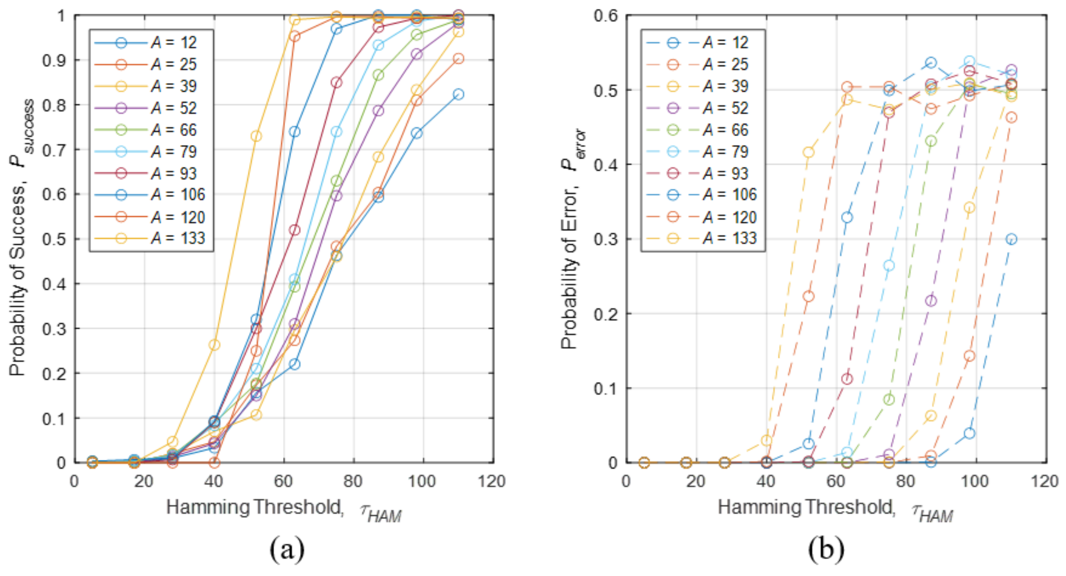


Figure 34. Hamming Method RNTI Recovery (a) $P_{success}$ and (b) P_{error} versus τ_{HAM} for $E = 256$, $SNR = 5$ dB, and $A =$ {12, 25, 39, 52, 66, 79, 93, 106, 120, 133}

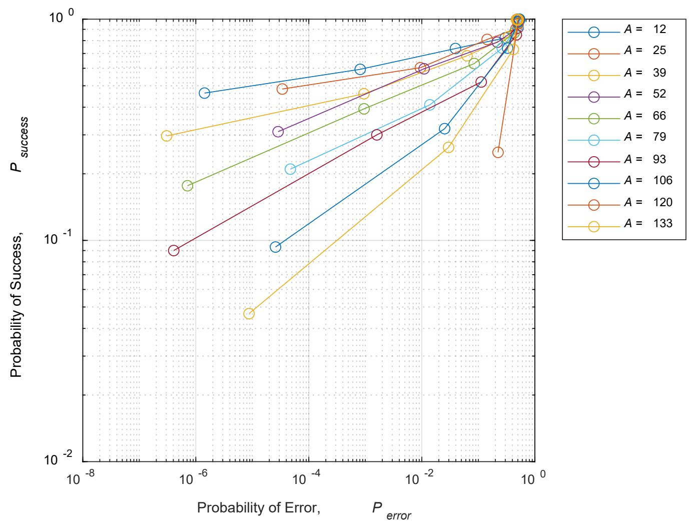


Figure 35. Hamming Method RNTI Recovery Efficiency for $E = 256$, $SNR = 5$ dB, and $A =$ {12, 25, 39, 52, 66, 79, 93, 106, 120, 133} as shown by $P_{success}$ versus P_{error}

To further investigate the $A = 120$ case, additional cases were simulated at $A =$ {116, 118, 119, 120, 121, 122, 124}, and the results are shown in Figure 36. We find that in this range only the specific payload and codeword combination of $E = 256$ and $A = 120$ has poor results. A possible explanation to this anomaly is that this is the result of a boundary case between rate matching methods. In Figure 37, it can be seen that $E = 256$ and $A = 120$ is at the boundary between the repetition, shortening, and no rate-matching methods. At $E = 256$ and $A = 120$, no rate-matching is applied because the codeword length is a factor of two and therefore matches with the polar coding block length N. The fact that different rate-matching techniques would be applied if the codeword was one bit longer or one bit shorter could be an indication that there is a unique effect in the polar coding or encoding process that causes this anomaly in RNTI recovery.

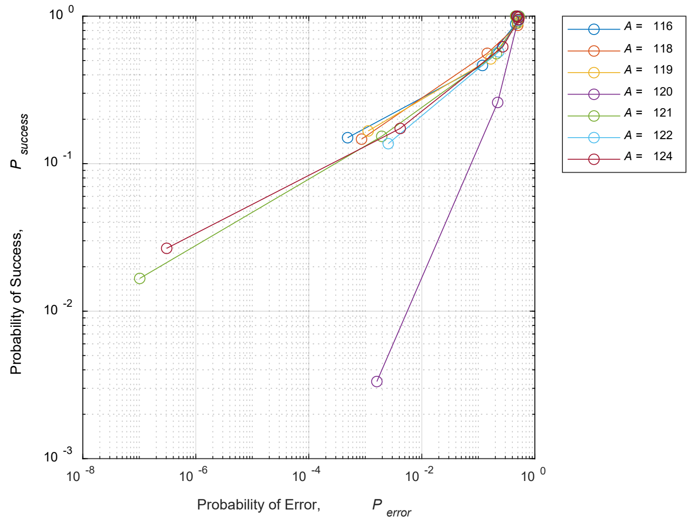


Figure 36. Hamming Method RNTI Recovery Efficiency for $E = 256$, $SNR = 5$, and $A = \{116, 118, 119, 120, 121, 122, 124\}$ as shown by $P_{success}$ versus P_{error}

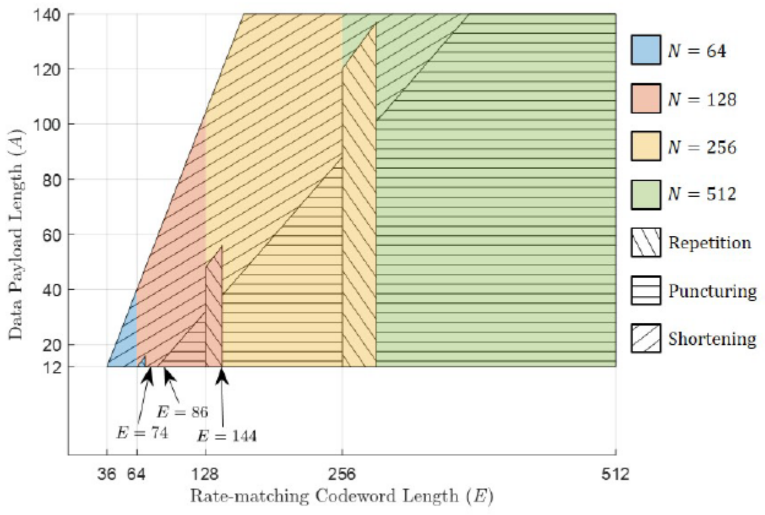


Figure 37. Rate-Matching Relationships between E and A in the PDCCH. Source: [3].

e. Codeword Length of 432

The codeword length $E = 432$ is expected to be used in practice and support all payloads as the maximum DCI payload for $P_{bler} \leq 10^{-3}$ at $SNR = 5$ dB is not limited by up to the maximum allowed $A = 140$ as found in Section IV.B. For $E = 432$, the payloads of $A =$ {12, 26, 40 55, 69, 83, 97, 112, 126, 140} were evaluated at τ_{HAM} = {5, 26, 48, 69, 90, 112, 133, 154, 176, 197}. From Figure 38, we see that $P_{success}$ and P_{error} follow the same monotonically increasing trends as for smaller codewords but here for the larger A values, and we see $P_{success}$ and P_{error} reach their maximums. As we compare $P_{success}$ and P_{error} to evaluate η_R, as shown in Figure 39, we continue to demonstrate that smaller A can recover RNTIs more efficiently for a fixed P_{error}.

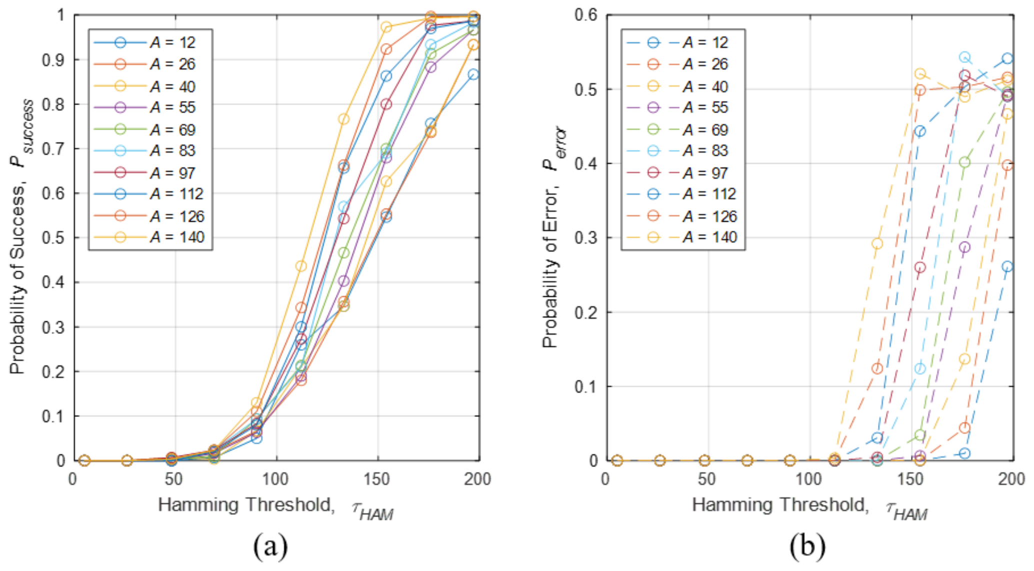


Figure 38. Hamming Method RNTI Recovery (a) $P_{success}$ and (b) P_{error} versus τ_{HAM} for $E = 432$, $SNR = 5$ dB, and $A = \{12, 26, 40, 55, 69, 83, 97, 112, 126, 140\}$

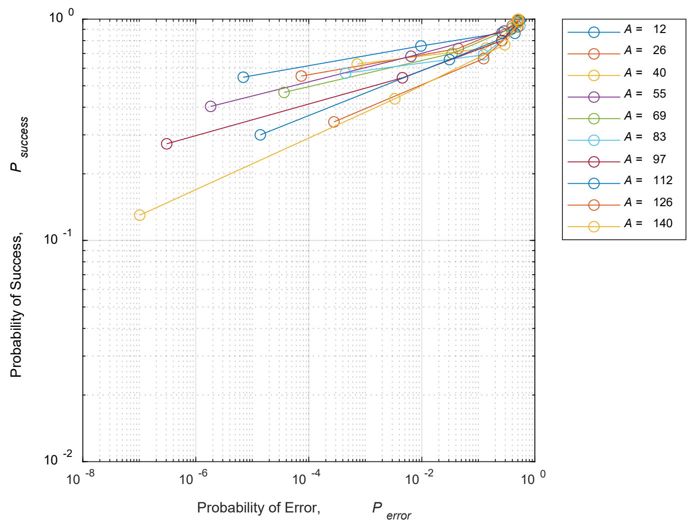


Figure 39. Hamming Method RNTI Recovery Efficiency for $E = 432$, $SNR = 5$ dB, and $A = \{12, 26, 40, 55, 69, 83, 97, 112, 126, 140\}$ as shown by $P_{success}$ versus P_{error}

f. Codeword Length of 512

The codeword length $E = 512$ is a theoretical case to match the polar coding block length of $N = 2^9 = 512$ and would support all payloads as found in Section IV.B. For $E = 512$, the payloads of $A =$ {12, 26, 40 55, 69, 83, 97, 112, 126, 140} were evaluated at $\tau_{HAM} = \{5, 31, 57, 83, 109, 134, 160, 186, 212, 238\}$. From Figure 40, we see that $P_{success}$ and P_{error} follow the same monotonically increasing trends as for smaller codewords but here for the larger A values, and we see $P_{success}$ and P_{error} reach their maximums. As we compare $P_{success}$ and P_{error} to evaluate η_R, as shown in Figure 41, we continue to demonstrate that smaller A can recover RNTIs more efficiently for a fixed P_{error}.

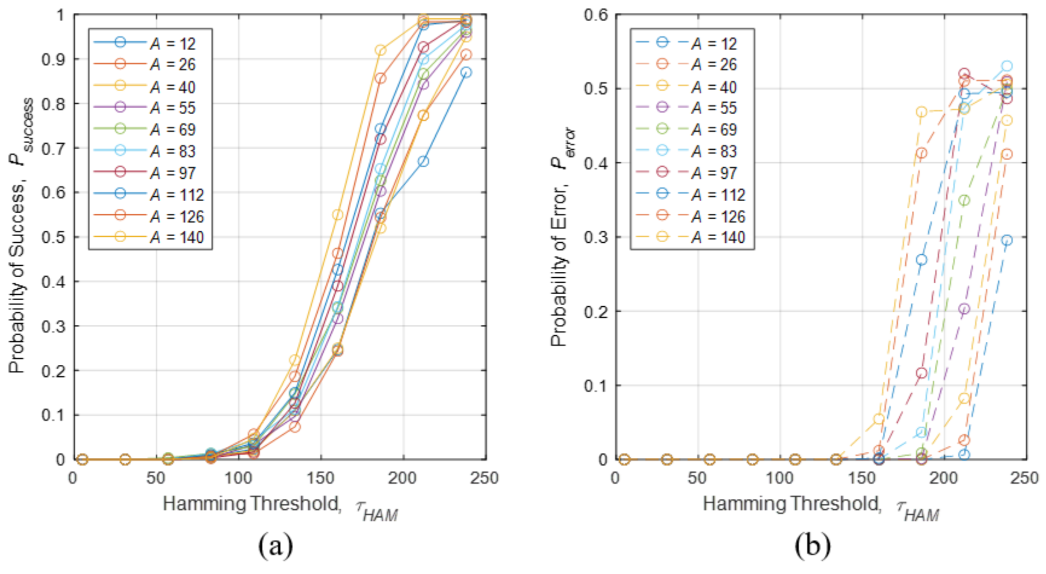


Figure 40. Hamming Method RNTI Recovery (a) $P_{success}$ and (b) P_{error} versus τ_{HAM} for $E = 512$, $SNR = 5$ dB, and $A = \{12, 26, 40, 55, 69, 83, 97, 112, 126, 140\}$

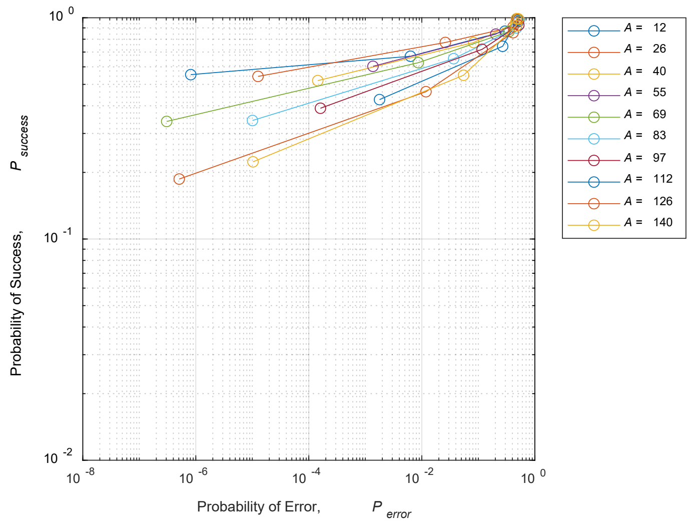


Figure 41. Hamming Method RNTI Recovery Efficiency for $E = 512$, $SNR = 5$ dB, and $A = \{12, 26, 40, 55, 69, 83, 97, 112, 126, 140\}$ as shown by $P_{success}$ versus P_{error}

3. Effect of Increased Codeword Length

As the codeword length E is expected to change based on the DCI message length A and the resources available in the PDCCH, this section evaluates the effect of increasing E on RNTI recovery probabilities. As E increases, the number of frozen bits F increases and as a result so does the length of the error pattern ε. Figure 42 shows the results of RNTI recovery simulations for a consistent payload of $A = 12$ across all codewords $E =$ {108, 128, 216, 256, 432, 512} using the τ_{HAM} values respective to each E as presented in Table 20. We note that in Figure 42 (a), $P_{success}$ increases monotonically with τ_{HAM}, but for larger codewords the slope of the relationship decreases since as ε increases, the variance increases according to $\sigma^2_{HAM} = \varepsilon / 4$. This means that for a larger codeword, a one-bit increase in τ_{HAM} will not cause as big of a change in the number of syndrome matches allowed through the threshold as for a smaller codeword, which is the biggest driving force of $P_{success}$. In Figure 42 (b), we observer that P_{error}, while low for the small $A = 12$, is nearly identical for all codewords but shifted as $\mu_{HAM} = \varepsilon / 2$ will affect the number of syndromes filtered.

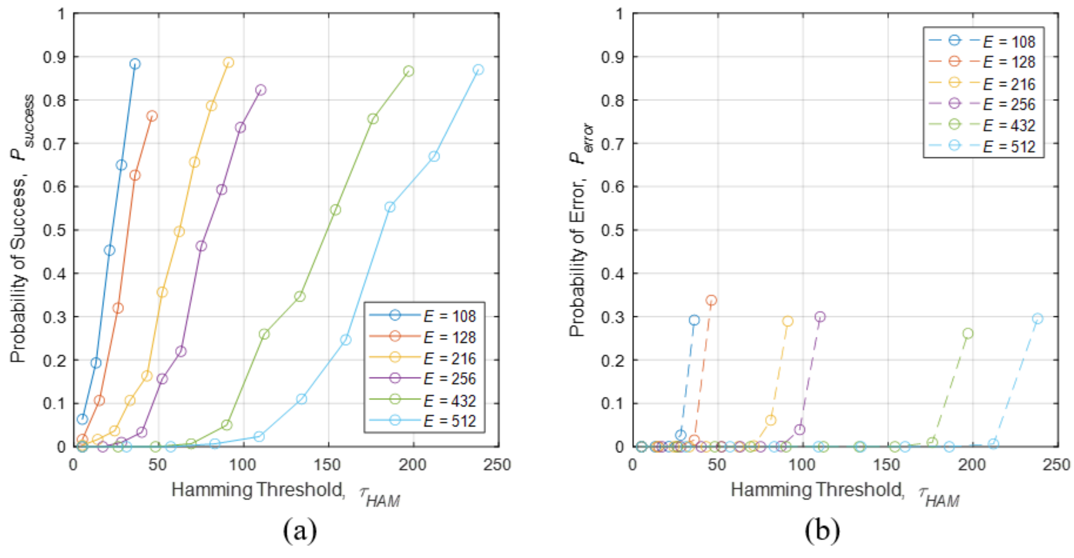


Figure 42. Hamming Method RNTI Recovery (a) $P_{success}$ and (b) P_{error} versus τ_{HAM} for $A = 12$, $SNR = 5$ dB, and $E =$ {108, 128, 216, 256, 432, 512}

We also consider the increase in RNTI recovery efficiency, η_R as E increases and the results are shown in Figure 43. As we have introduced in Section III.C.2, with larger F and ε, we can recover RNTIs more efficiently due to the higher code rate and a less likelihood of the correct RNTI being rejected by the threshold. Figure 43 shows that as E increases, η_R increases as well. The results are not as distinctive for codewords close in size, but we can clearly see that $E = \{432, 512\}$ demonstrate higher η_R than $E = \{108, 128\}$. As we consider a gNB choosing a codeword, the benefit of a larger E is that it has a higher code rate, but the downsides are that more PDCCH resources are expended and the RNTI can more easily and more efficiently be recovered by an attacker. Lastly, in Figure 44, we compare the effect of increasing codeword has on η_R across all E, A, and τ_{HAM} evaluated in this thesis.

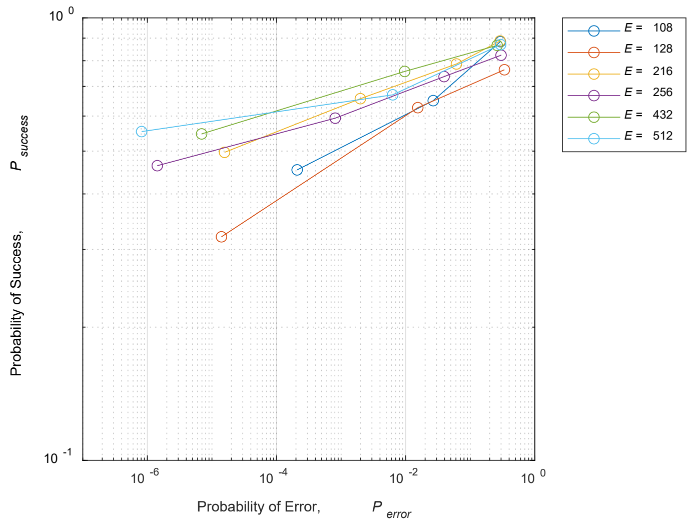
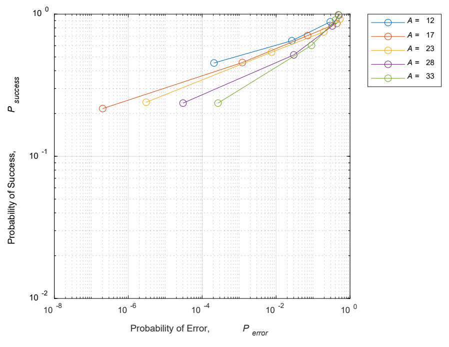
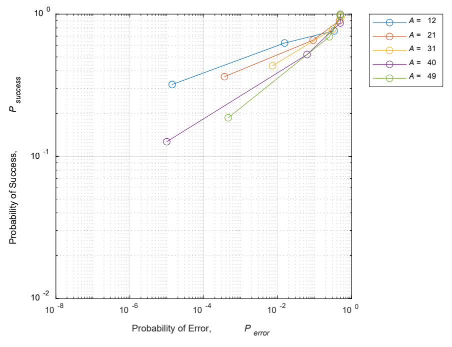


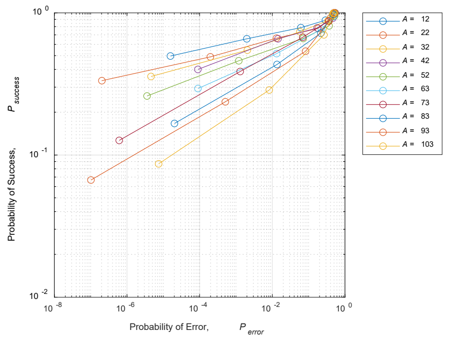
Figure 43. Hamming Method RNTI Recovery Efficiency for $A = 12$, $SNR = 5$ dB, and $E = \{108, 128, 216, 256, 432, 512\}$ as shown by $P_{success}$ versus P_{error}



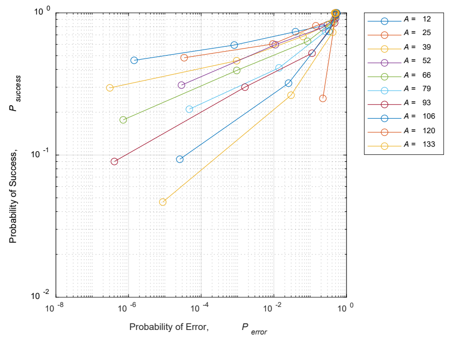
(a) $E = 108$



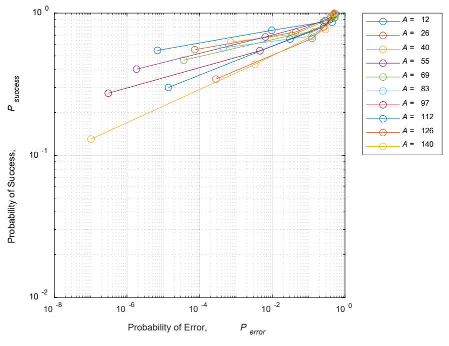
(b) $E = 128$



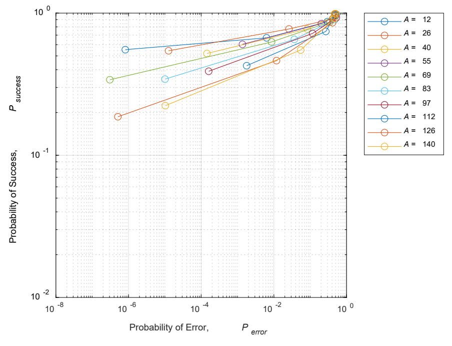
(c) $E = 216$



(d) $E = 256$



(e) $E = 432$



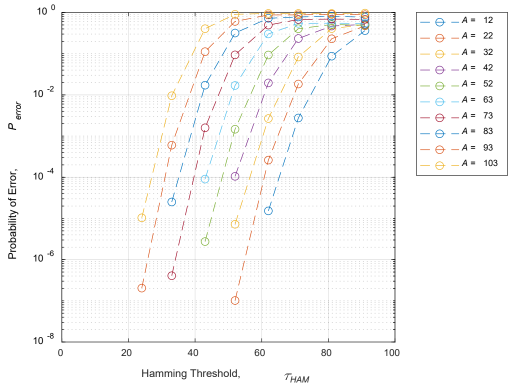
(f) $E = 512$

Figure 44. Hamming Method RNTI Recovery Efficiency for $SNR = 5$ dB, $E = \{108, 128, 216, 256, 432, 512\}$ and Set Ranges of A as shown by $P_{success}$ versus P_{error}

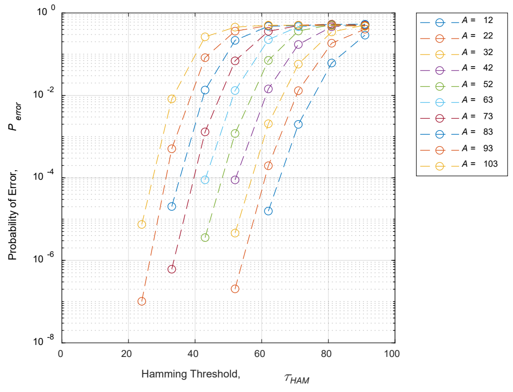
4. Effect of High and Low SNR

As discussed in Section III.C.5, we expect an attacker to experience a different SNR than the UE, and therefore we evaluate effect of SNR on the RNTI recovery simulation in this section. We choose a base case of $E = 216$ using the A and τ_{HAM} values established for this codeword in Table 20 and used previously in Section IV.D.2.c. First, we evaluate the effect of SNR on P_{error} and the results are shown in Figure 45 (a) $SNR = 0$ dB, (b) $SNR = 5$ dB, and (c) $SNR = 10$ dB. The results are identical with some minor differences at very low P_{error} that can be attributed to the limited number of trials. We conclude that due to the random nature of the RNTI initiated scrambling and the large number of syndomes (2^{15}), P_{error} is not affected by the SNR.

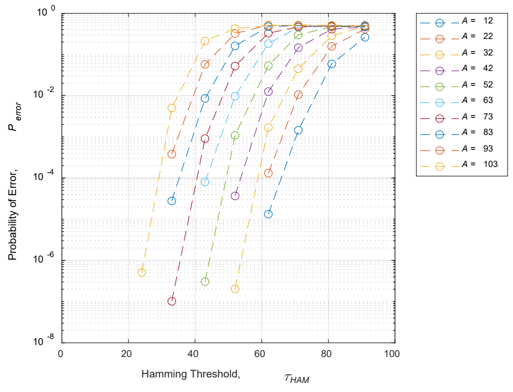
Next, we evaluate the effect of of SNR on $P_{success}$, and the results are shown in Figure 46 (a) $SNR = 0$ dB, (b) $SNR = 5$ dB, and (c) $SNR = 10$ dB. Here, as expected, we find a significant increase in $P_{success}$ as SNR increases. We recall that a higher SNR will lead to a lower probability of channel bit error $P_{b,QPSK}$ and recalling Figure 11 will have a lower number of channel bit errors. Further, once the correct error pattern is filtered to be checked for CRC, the subsequent low P_b also means there is a low probability that the CRC will be corrupted. In conclusion, these two factors lead to a high probability that the RNTI will be recovered succesfully at high $SNR = 10$ dB case as validated by the results. For the low $SNR = 0$ dB case, the opposite effect is expected and we actually observe a $P_{success}$ limit where there are uncorrectable errors such that even as we increase τ_{HAM} and move towards the brute force case, the probability that we match the CRC is limited. We observe limits for $A =$ {73, 83, 93 103} of $P_{success} \approx$ {0.65, 0.50, 0.25, 0.15} where this is the case. Finally, we recall that $\eta_R \propto \frac{P_{success}}{P_{error}}$ and since $P_{success}$ increases significantly for increasing SNR while P_{error} remains constant, we expect η_R to also increase significantly for increasing SNR. These results are shown in Figure 47 for (a) $SNR = 0$ dB, (b) $SNR = 5$ dB, and (c) $SNR = 10$ dB.



(a) $SNR = 0$ dB

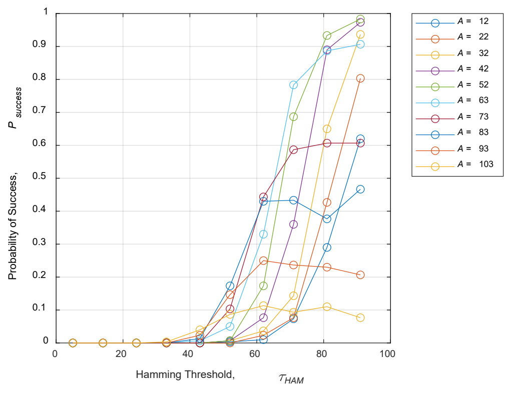


(b) $SNR = 5$ dB

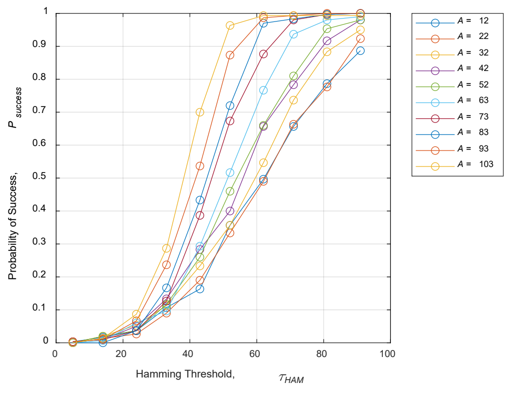


(c) $SNR = 10$ dB

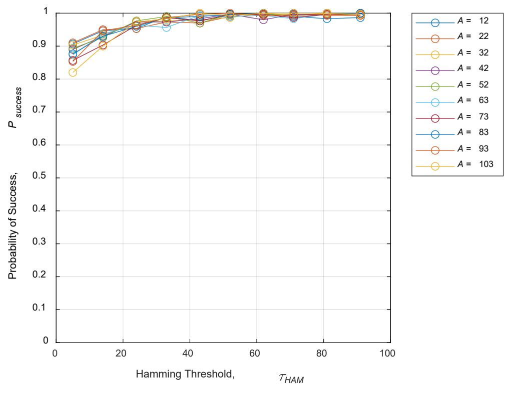
Figure 45. Hamming Method RNTI Recovery P_{error} versus τ_{HAM} for $E = 216$ and $A = \{12, 22, 32, 42, 52, 63, 73, 83, 93, 103\}$



(a) $SNR = 0$ dB

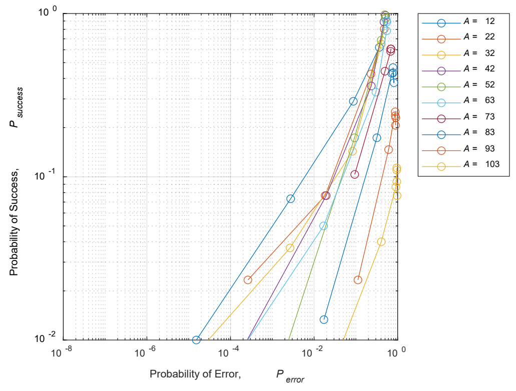


(b) $SNR = 5$ dB

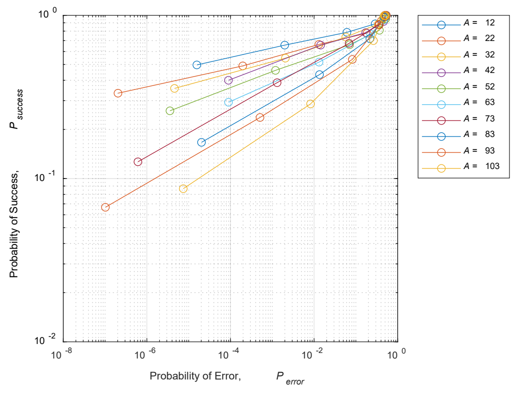


(c) $SNR = 10$ dB

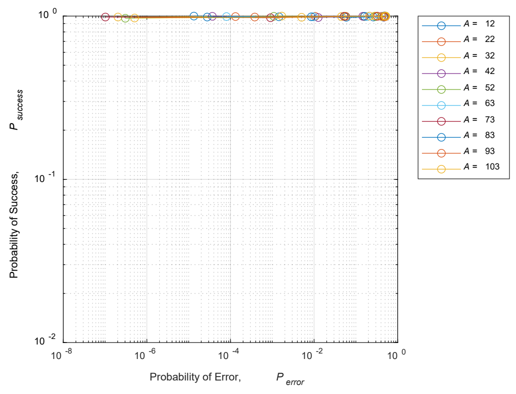
Figure 46. Hamming Method RNTI Recovery $P_{success}$ versus τ_{HAM} for $E = 216$ and $A =$ {12, 22, 32, 42, 52, 63, 73, 93, 103}



(a) $SNR = 0$ dB



(b) $SNR = 5$ dB



(c) $SNR = 10$ dB

Figure 47. Hamming Method RNTI Recovery Efficiency for $E = 216$ and $A =$ {12, 22, 32, 43, 52, 63, 73, 83, 93, 103} as shown by $P_{success}$ versus P_{error}

5. Effect of Recovery Method (LCS versus Hamming)

In this section, we evaluate using the LCS method to determine whether error patterns match syndromes within a τ and compare the results to the Hamming method. The LCS method calculates the longest common substring of matching bits between the two and we determined in Section II.C.6 that we can use $\mu_{LCS} = 3\log(\varepsilon)$ to estimate the median m_{LCS}, which serves as a starting τ_{LCS} such that $P_{error} \approx 0.50$. Increasing τ_{LCS} has an opposite effect of an increase of τ_{HAM} as for the LCS method fewer error patterns will match substrings of greater lengths with syndromes, thus the matching filter becomes more restrictive.

a. LCS Baseline Case

We first consider our baseline case of $E = 216$ with $A = \{12, 22, 32, 42, 52, 63, 73, 83, 93, 103\}$ and $\tau_{LCS} = \{5, 10, 14, 24, 33, 43, 52, 62, 71, 81, 90\}$. We have generated the τ_{LCS} range by modifying the τ_{HAM} range, adding a threshold of $\tau_{LCS} = 10$ to increase the fidelity at low thresholds as d_{LCS} will be lower on average than d_{HAM}. The results of the LCS method for $E = 216$ are presented in Figure 48 where we observe the monotonically decreasing behavior for $P_{success}$ and the exponentially decreasing behavior for P_{error}. We note that for the LCS method, P_{error} does not change signficiantly over the range of A and τ_{LCS}, which is due to the fact that the mean estimate $\mu_{LCS} = 3\log(\varepsilon)$ does not change significantly for small changes in ε. Lastly, we demonstrate the η_R results in Figure 49, and we observe the same behavior as the Hamming method in that we can recover RNTIs more efficiently for smaller A due to the higher code rate and higher likelihood of being filtered correctly due to the larger ε.

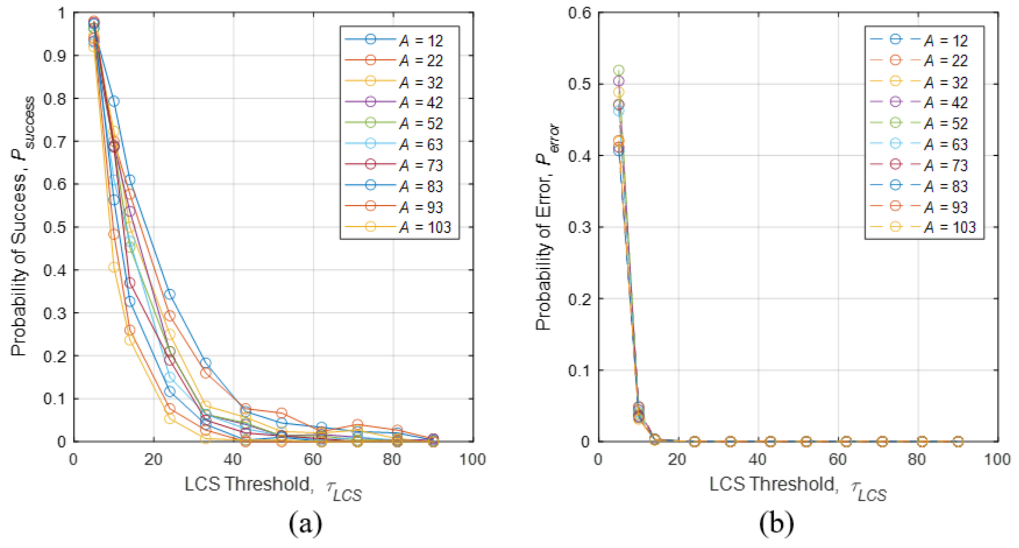


Figure 48. LCS Method RNTI Recovery (a) $P_{success}$ and (b) P_{error} versus τ_{LCS} for $E = 216$, $SNR = 5$ dB, and $A = \{12, 22, 32, 42, 52, 63, 73, 83, 93, 103\}$

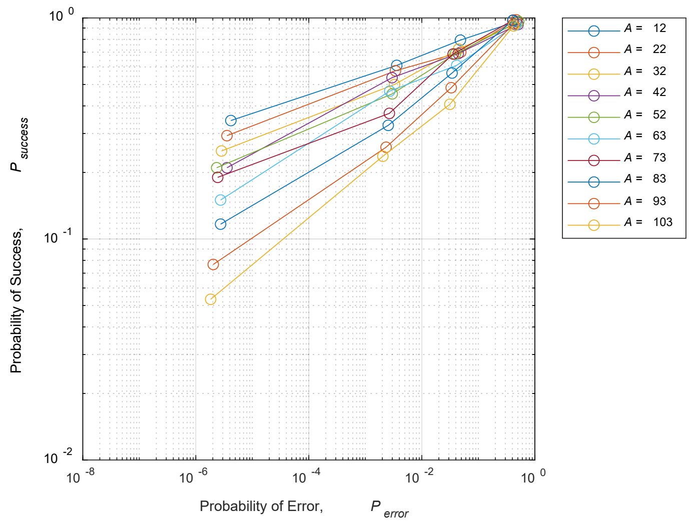
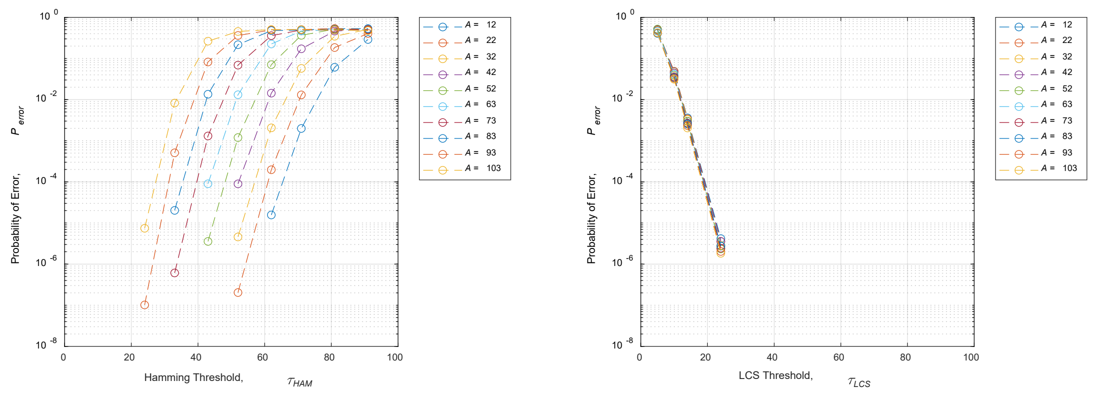


Figure 49. LCS Method RNTI Recovery Efficiency for $E = 216$, $SNR = 5$ dB, and $A = \{12, 22, 32, 42, 52, 63, 73, 83, 93, 103\}$ as shown by $P_{success}$ versus P_{error}

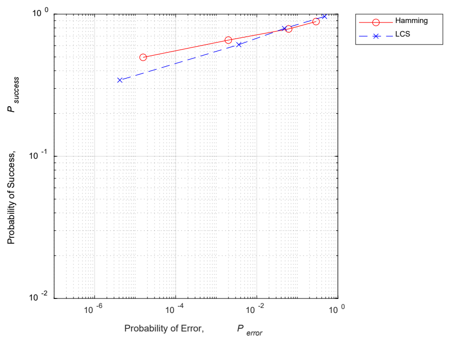
We want to further explore the difference in P_{error} versus τ between the Hamming and LCS methods, and we show the results for comparison in Figure 50 (a) Hamming method and (b) LCS method. As we observe in Figure 50 (b), the LCS method has an advantage in that P_{error} does not change signficiantly over the range of A and τ_{LCS}. The significance of this result is that an attacker running a RNTI recovery process would not need to adjust τ_{LCS} for different A within a recovered E. Considering that the attacker will know E from the number of QPSK symbols intercepted but not A makes this result even more substantial.



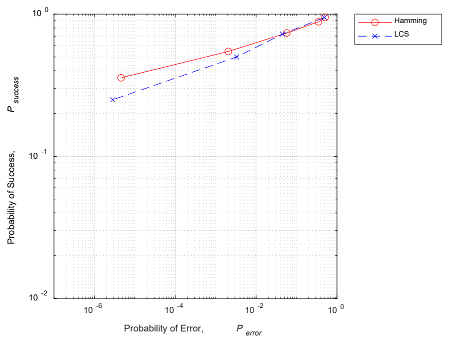
(a) Hamming Method (b) LCS Method

Figure 50. RNTI Recovery P_{error} versus τ for $E = 216$, $SNR = 5$ dB, and $A =$ {12, 22, 32, 42, 52, 63, 73, 83, 93, 103}

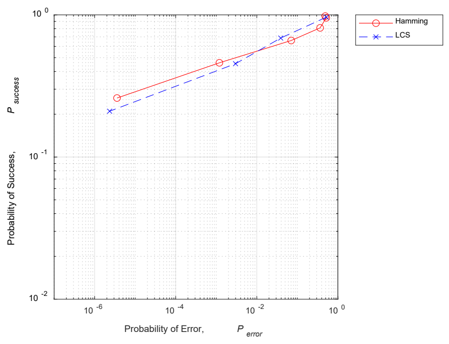
Finally, we compare η_R for $E = 216$ and $SNR = 5$ dB between the Hamming and LCS methods to determine if one is more efficient than the other in recovering RNTIs. This result is best illustrated individually for select A as shown in Figure 51. We conclude that neither Hamming nor LCS method presents a decisive advantage in η_R at $SNR = 5$ dB.



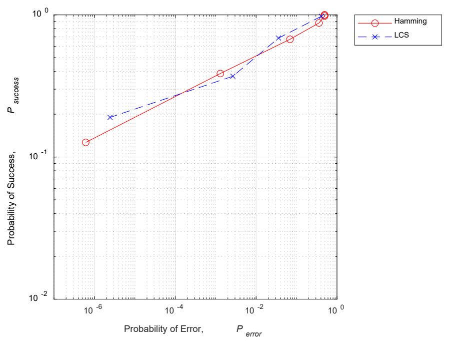
(a) $A = 12$



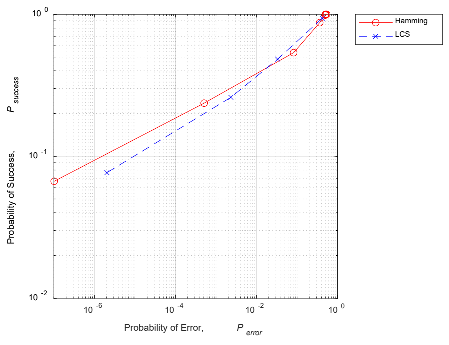
(b) $A = 32$



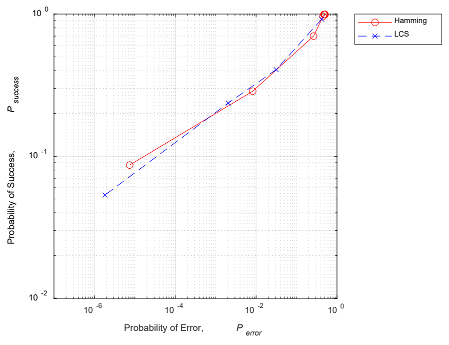
(c) $A = 52$



(d) $A = 73$



(e) $A = 93$

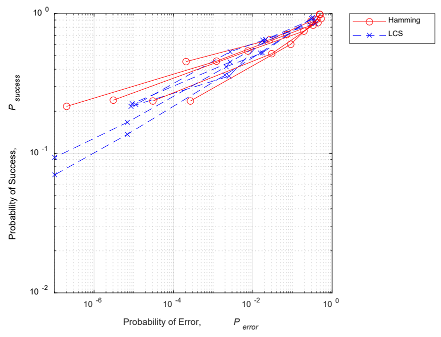


(f) $A = 103$

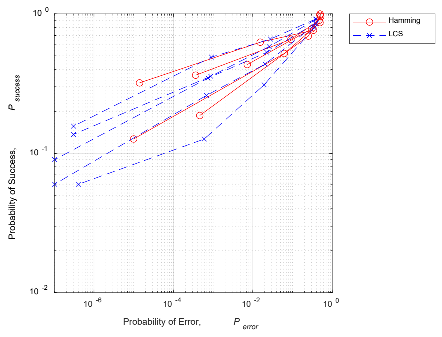
Figure 51. Hamming and LCS RNTI Recovery Efficiency for $E = 216$ and $SNR = 5$ dB at Selected Payloads as shown by $P_{success}$ versus P_{error}

b. LCS versus Hamming Across All Codewords

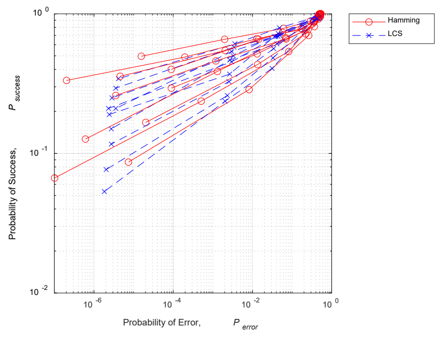
Next, we expand to evaluate the LCS method across all codewords $E = \{108, 128, 216, 256, 432, 512\}$ and payloads A as specified for each codeword in Table 20. The results are shown in Figure 52, and we observe that the Hamming method becomes slightly more effective than LCS at $E = \{432, 512\}$. While we do not have any definitive theory as to why this is the case, we postulate that it may have to do with the fact that as ε increases, more matching substrings can exist within these longer error patterns, which could result in an increase in P_{error}. We note that the $A = 120$ and $E = 256$ case, as can be identified in Figure 52 (d) with the outlier lines to the far right of the plot, has the same anomaly in $P_{success}$ in the LCS method as was found earlier in the Hamming method. This further backs the hypothesis that the anomaly has to do with the PDCCH encoding and decoding process, possibly the polar coding, and is not associated with the recovery method.



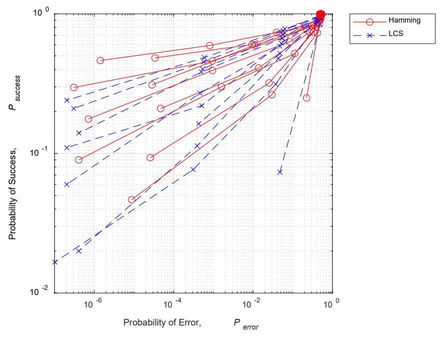
(a) $E = 108$



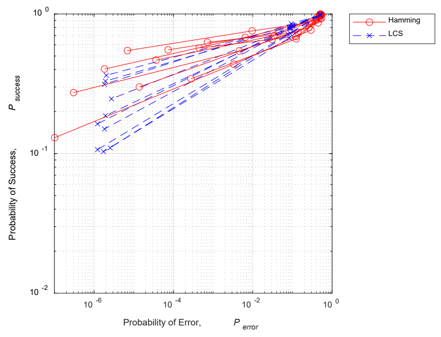
(b) $E = 128$



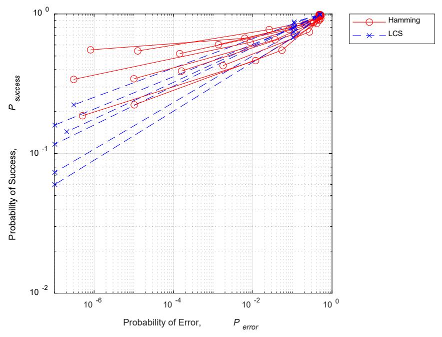
(c) $E = 216$



(d) $E = 256$



(e) $E = 432$

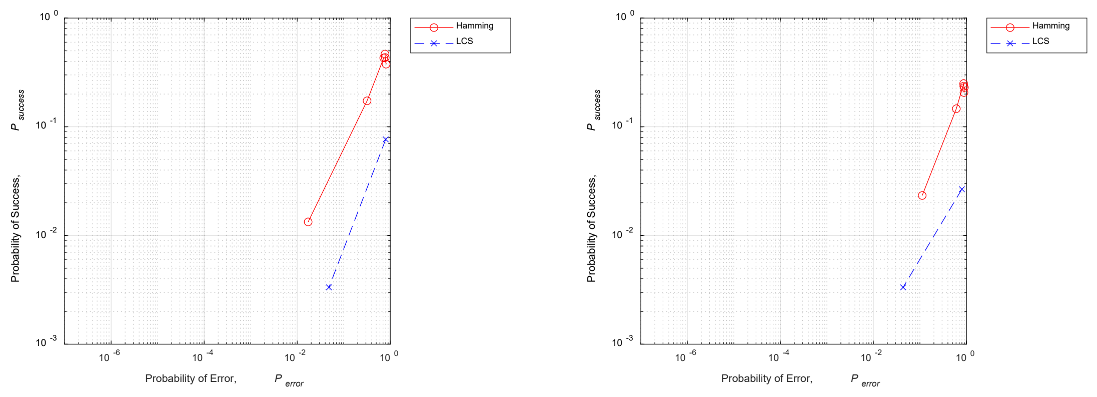


(f) $E = 512$

Figure 52. Comparison of Hamming and LCS RNTI Recovery Efficiency Across All Payloads and Codewords as specified in Table 20 as shown by $P_{success}$ versus P_{error}

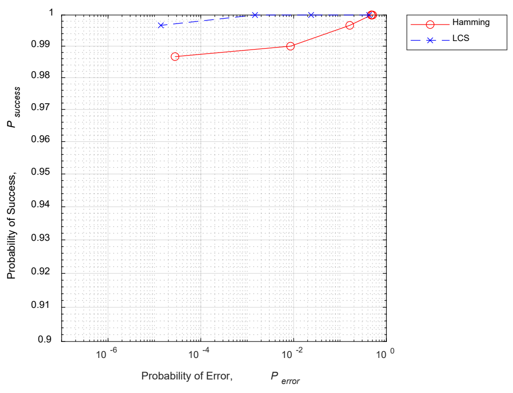
c. LCS versus Hamming at High and Low SNR

We conclude our LCS analysis by considering any advantages of LCS and Hamming methods at high or low SNR. In [3], it was found that the Hamming method had a slight advantage at the lower $SNR = 5$ dB while the LCS method had a slight advantage at the higher $SNR = 8$ dB when limiting $P_{error} \le 10^{-4}$. In this thesis, we consider much higher P_{error} as a practical setup is expected to have reasonable computational power to pursue a high $P_{success}$ at the cost of high P_{error}. We consider our baseline case of $E = 216$ with $A = \{12, 22, 32, 42, 52, 63, 73, 83, 93, 103\}$ and $\tau_{LCS} = \{5, 10, 14, 24, 33, 43, 52, 62, 71, 81, 90\}$, and we show selected results in Figure 53 for $SNR = 0$ dB and Figure 54 for $SNR = 10$ dB. We find that for $SNR = 0$ dB there is a slight advantage to the Hamming method, and for $SNR = 10$ dB there is a slight advantage to the LCS method as predicted and shown for one case in [3]. These advantages are most prevalent at the higher A as fewer error pattern bits lead to more constrained matching.

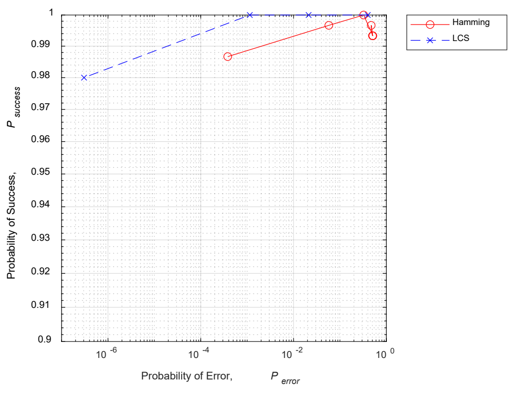


(a) (b)

Figure 53. Comparison of Hamming and LCS RNTI Recovery Efficiency for $E = 216$ and $SNR = 0$ dB at (a) $A = 83$ (b) $A = 93$



(a)



(b)

Figure 54. Comparison of Hamming and LCS RNTI Recovery Efficiency for $E = 216$ and $SNR = 10$ dB for (a) $A = 83$ and (b) $A = 93$

E. ACTIVITY RECOVERY DISCUSSION

Overall, the goal of the RNTI recovery process is to de-anonymize the PDCCH channel and recover activity of the UEs. Traffic analysis and information from sources other than intercepted PDCCH messages are required to match RNTIs to UEs. First, new devices connecting or re-connecting to a gNB present an opportunity to link a UE to a RNTI if a new RNTI is recovered at the same time. Second, measuring the frequency of PDCCH messages sent to a specific RNTI and associating UE activity with an increase in messages can be used to link the two. In many cases, the association of a UE with a RNTI requires another information source such as visual confirmation of the UE entering the area or another selector that reveals the UE is performing an action to increase PDCCH messages. One such possibility is to link the TA recovered from the MAC CE on the PDSCH to the recovered and associated RNTI. It has been shown that for 5G numerologies the TA can be used to localize a UE with a 95% circular error probability on the order of 10-100 m depending on the numerology, number of remote radio heads, and beamforming applications [25], [26].

Once RNTI and UE associations are established, the goal is to monitor PDCCH messages for changes in UE activity. Once a RNTI is recovered, all PDCCH DCI messages can be decoded, and the associated parameters recovered. The DCI parameters can be tracked to identify changes that would reveal a change in geographic location, change in mode, or the use of an application by a tracked UE as presented in Section III.D.2. In Figure 55, a concept of RNTI-UE relationship and activity tracking is envisioned where recovered RNTIs, PDCCH message rates, recovered DCI information revealing directed power, mode of operation, and MCS, and any other recovered information are monitored for a mobile environment.

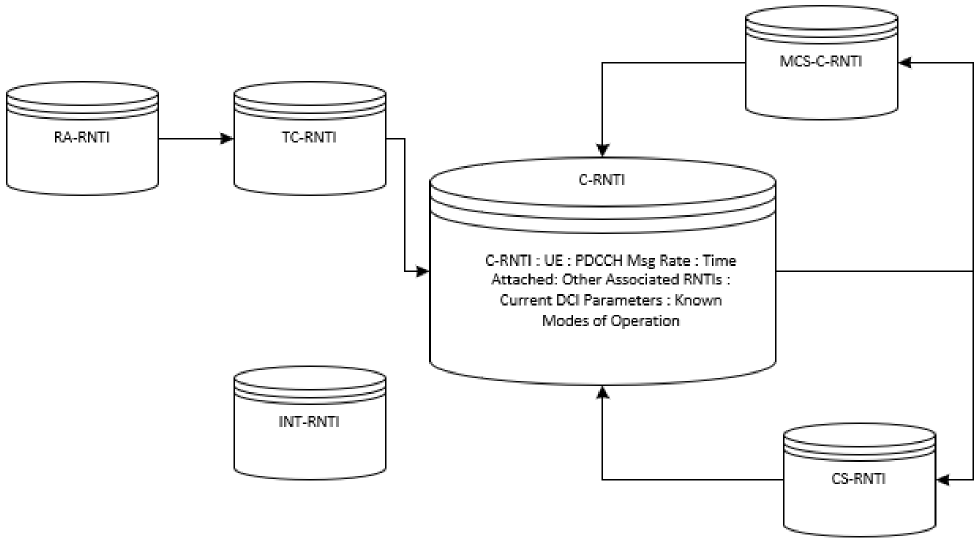

RA-RNTI
TC-RNTI
MCS-C-RNTI
C-RNTI
C-RNTI : UE : PDCCH Msg Rate : Time Attached: Other Associated RNTIs : Current DCI Parameters : Known Modes of Operation
INT-RNTI
CS-RNTI

Figure 55. Concept for PDCCH Traffic Analysis of Known UE-RNTI Relationships and Activity, as Derived from Recovered RNTIs and Decoded DCI Messages

In summary, we presented the simulation model and results of the maximum DCI payload and RNTI recovery processes. We evaluated how τ, A, E, and the recovery method impact $P_{success}$, P_{error}, and η_R. We validated our assumptions and statistical derivations that ε, which changes with A and E, is a critical driver of $P_{success}$, P_{error}, and η_R. Further, we showed that SNR can drastically change the $P_{success}$ and η_R of RNTI recovery and should be maximized by an attacker. We concluded that the LCS method can be used more easily by an attacker looking to hold computational resources constant through a consistent P_{error}. Finally, we presented a basic strategy to use decoded DCI messages and RNTI-UE traffic analysis to track UEs operating in a mobile environment.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS

In this thesis, we have evaluated the ability for an attacker to recover RNTIs and reveal user activity from the DCI messages and RNTIs in the PDCCH. We considered the impact of the threshold, payload size, codeword size, high and low SNR, and Hamming versus LCS methods. We performed statistical analysis of the Hamming and LCS methods to assist in choosing the optimal threshold to meet the needs of the mobile environment and the attacker's RNTI recovery goals. We also considered what probable DCI lengths would be used in the PDCCH and how to RNTI recovery thresholds based on the P_{bler}, SNR, and E. Finally, we presented the $P_{success}$, P_{error}, and η_R of RNTI recovery and proposed a model for tracking user activity through RNTI recoveries.

A. CONTRIBUTIONS

Through the statistical analysis of data from the Hamming and LCS methods, we developed probability distributions for the expected Hamming distance and LCS distance. We found that the Hamming distance follows a Gaussian distribution with $\mu_{HAM} = \varepsilon / 2$ and $\sigma^2_{HAM} = \varepsilon / 4$, where ε is the length of the sequence; for LCS, μ_{LCS} and σ^2_{LCS} increase logarithmically as ε increases and specifically $\mu_{LCS} \approx 3\log(\varepsilon)$ for $\varepsilon < 100$. Further, we determined how a channel bit error can propagate through the modified polar decoding step in RNTI recovery and presented the expected Hamming and LCS outputs for input channel bit errors ranging from one to ten.

When comparing RNTI recovery threshold methods, we found that the LCS method can be more optimal for a real-world simulation in that the threshold will not need to be adjusted for changes in payload length. The Hamming method, however, can be more finely tuned when compared to the LCS method. Further, we confirmed for a case of $E = 216$ that the Hamming method has a slight advantage at low SNR while the LCS method has a slight advantage at high SNR.

For the RNTI recovery methodology, we presented results on $P_{success}$, P_{error}, and η_R across a myriad of payloads, codewords, SNR s and recovery methods to demonstrate that a decrease in payload length, an increase in codeword length, and high SNR all improve RNTI recovery efficiency η_R. Further, we considered the activity of a UE that could be recovered from the decoded DCI messages and presented a model in which recovered RNTIs and information recovered from DCI fields can be used to determine the change in geographic location or the change in operating mode of a UE. This model shows the real-world possibilities of utilizing the RNTI recovery methodology to track UEs through a 5G network.

B. FUTURE WORK

The methods presented in this thesis can certainly be put into practice to recover RNTIs in a simulated PDCCH channel. In a physical experiment, the PDCCH channel would be monitored for codewords, and these would be intercepted and processed to recover RNTIs. This experiment could be used to validate the assumptions used in assessing that the activity of a UE if changing location or changing mode of operation could be deduced from intercepted DCI messages. In the experiment, a UE can be established on a PDCCH with the gNB and then the UE can be moved further or closer to the gNB to monitor the change in DCI messaging associated with the change in geographic location. Further, the experiment could be repeated for changing mode of operation to URLCC or to an application utilizing a CS-RNTI and CS-associated messaging, such as VOIP. Finally, the experiment could test the optimization methods by adjusting threshold values to validate the RNTI recovery probabilities and efficiencies.

Further work first identified in [3], considers the application of the RNTI recovery model and optimization to the PUCCH, which similarly utilizes RNTI initiated scrambling and polar coding thus is vulnerable to the same methods. While the PDCCH certainly is more enticing in terms of the information recovered, the PUCCH contains HARQ acknowledgments, scheduling requests, and CSI reports from the UE. For a sophisticated attacker, a RNTI recovery model that recovers both the PDCCH and PUCCH could use the two to gain a comprehensive understanding of the mobile environment and the UEs operating in it.

While this thesis evaluated the RNTI recovery $P_{success}$, P_{error}, and η_R for various E and A combinations to include rate-matching cases, the impact of rate-matching was not specifically evaluated during this study. A future work could refer to [3] for further details on the effect of rate matching on modified polar decoding and syndrome matching and compare and contrast the RNTI recovery statistics for the different rate matching cases. The $E = 256$ and $A = 120$ boundary case would certainly be of interest, and it is postulated that there are other boundary cases and/or optimization cases for which the RNTI recovery statistics exceed the expected values due to rate-matching effects.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. MAXIMUM DCI PAYLOAD MODEL MATLAB CODE

The MATLAB code in this Appendix was developed using the MATLAB 5G toolbox to simulate the encoding and modulation of PDCCH messages in a AWGN channel, adapted in part from [25]. In this code, we encode a random message of length A onto a block of length E in accordance with the 3GPP standard, modulate the bits to QPSK symbols, and add AWGN according to a specified SNR. We then receive, demodulate, and decode a recovered message. If bit errors occur between the recovered message and the sent message, then a block error is recorded. The process is repeated for a large number of trials at different A, E and the results of P_{bler} are presented.

```
clearvars
clear all
```

Input Testing Info

```
%Input Testing Info
teststotal=10
minpayload=12
maxpayload=140
codewords=[108 128 216 256 432 512]

%initialize output data vectors
bler=zeros(1,maxpayload);
errorlog=zeros(teststotal,maxpayload);
trackerBLER=zeros(length(codewords),maxpayload);
codetrack=1;

for codeword=codewords
for payload=minpayload:1:maxpayload
payload
tic
for tests=1:teststotal
```

System Parameters

```
nID = 10;           % pdcch-DMRS-ScramblingID
rnti = randi([1 65519]);         % C-RNTI for PDCCH in a UE-specific
search space
A = payload;                % Number of DCI message bits
E = codeword;               % Number of bits for PDCCH resources
if E-A>24 %break if payload too large
```

DCI Encoding

```
dciBits = randi([0 1],A,1,'int8');
dciCW = nrDCIEncode(dciBits,rnti,E);
```

PDCCH Symbol Generation

```
sym = nrPDCCH(dciCW,nID,rnti);
```

Channel

```
snrdB=5;
rxSym = awgn(sym,snrdB,'measured');
```

PDCCH Decoding

```
noiseVar = 10.^(-snrdB/10);     % assumes unit signal power
rxCW = nrPDCCHDecode(rxSym,nID,rnti,noiseVar);
```

DCI Decoding

```
listLen = 1;                      % polar decoding list length
[decDCIBits,mask] = nrDCIDecode(rxCW,A,listLen,rnti);

isequal(mask,0);
blockdecoded=isequal(decDCIBits,dciBits);
```

Error Calculation

```
errorlog(tests,payload)=blockdecoded;
else
errorlog(tests,payload)=NaN;
end
end %end tests loop
toc
errors=size(errorlog(errorlog(:,payload)==0),1);
total=errors+size(errorlog(errorlog(:,payload==1)),1);
```

```
if E-A>24
    bler(payload)=errors/total;
else
    bler(payload)=NaN;
end
end %end payload loop

trackerBLER(codetrack,:)=bler;
codetrack=codetrack+1
end %end codeword loop


%Plot BLER versus Payload Size
figure()
plot([1:1:maxpayload],trackerBLER)
ylabel('Block Error Rate')
xlabel('Payload Size (bits)')
grid on

%Plot BLER versus Payload Size (semilog)
figure()
semilogy([1:1:maxpayload],trackerBLER)
ylabel('Block Error Rate')
xlabel('Payload Size (bits)')
grid on
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. RNTI RECOVERY MODEL MATLAB CODE

The MATLAB code in this Appendix is adapted from [3], which was designed to demonstrate the RNTI recovery methodology for $A = 12$, $E =$ {74, 86, 128, 144}. In this code, we expand the model to cover the entire range of $A =$ [12:140] and all possible E. Further, this code incorporates the MATLAB 5G functions in lieu of some functions manually developed in [3]. Overall, this MATLAB model simulates the encoding and modulation process, modified polar decoding with syndrome matching, and RNTI recovery, and presents the $P_{success}$, P_{error}, and η_R results.

```
clearvars
clear all
```

1. Define Testing Variables and Loops

```
startmodel=tic;
%set variable limits for testing
payloadlengths=[12 22 32 42 52 63 73 83 93 103] %set payload(s) to be
analyzed
tests=10    %set number of tests
% set hamming or lcs threshold methodlogy in Section 4, limits below
thresholdvalues=[5 14 24 33 43 52 62 71 81 91]

%Testing variables
threshtotalcounter=1; payloadcounter=1;

%initialize trackers for code speed
numpayloads=length(payloadlengths);
numthreshs=length(thresholdvalues);
tracker=zeros(numpayloads*numthreshs,6);
type1errors=zeros(numpayloads,numthreshs);
successes=zeros(numpayloads,numthreshs);
RNTIsmatchedavg=zeros(numpayloads,numthreshs);

%For loop for each payload length
for payload=payloadlengths
threshcounter=1;
%initialize tracking variables for matches and errors
```

```
match=0; nomatch=0; thresh=0; false=0; threshold_matches=0;

%For loop for each threshold value
for threshold=thresholdvalues
tic

%For loop for number of tests at each payload, threshold
for testcounter=1:tests
```

2. Define DCI message parameters

```
E = 216; % rate-matching codeword length
nRNTI = randi([1 65519]); % rnti value, legal range is [1 65519]
nID = 10; % cell scrambling ID
a = randi([0 1],payload,1); % data payload
snr = 5; % define SNR level for channel
```

3. PDCCH encoding

```
dciCW=nrDCIEncode(a,nRNTI,E); %encode DCI message
sym = nrPDCCH(dciCW,nID,nRNTI); %generate QPSK symbol
rxSym = awgn(sym,snr,'measured'); %add AWGN
```

4. RNTI Recovery

Demodulate

```
rxLLR = nrSymbolDemodulate(rxSym,"QPSK"); % demodulate received symbols
to LLR values
```

Determine frozen bit set qF and information bit set qI

```
% Get N
K=length(a)+24; %add CRC bits
nMax = 9;       % maximum n value for N
N = nr5g.internal.polar.getN(K,E,nMax);

% Get sequence for N, ascending ordered
s10 = nr5g.internal.polar.sequence;          % Nmax=10
idx = (s10 < N);
qSeq = s10(idx);                            % 0-based

% Get frozen, information bit indices sets, qF, qI
jn = nr5g.internal.polar.subblockInterleaveMap(N);  % 0-based
qFtmp = []; qFPunc = [];
```

```
if E < N
    if K/E <= 7/16  % puncturing
        for i = 0:(N-E-1)
            qFPunc = [qFPunc; jn(i+1)];   % punctured bits
        end
        if E >= 3*N/4
            uLim = ceil(3*N/4-E/2);
            qFtmp = [qFPunc; (0:uLim-1).']; % extra freezing
        else
            uLim = ceil(9*N/16-E/4);
            qFtmp = [qFPunc; (0:uLim-1).']; % extra freezing
        end
        qFtmp = unique(qFtmp);
    else            % shortening
        for i = E:N-1
            qFtmp = [qFtmp; jn(i+1)];   % shortened bits
        end
    end
end

% Get qI from qFtmp and qSeq
qI = zeros(K,1);
j = 0;
for i = 1:N
    ind = qSeq(N-i+1);      % flip for most reliable
    if any(ind==qFtmp)
        continue;
    end
    j = j+1;
    qI(j) = ind;
    if j==(K)
        break;
    end
end

% Form the frozen bit vector
qF = setdiff(qSeq,qI);     % sorted doesn't matter now
```

Rate-recovery

```
if E == N % no rate matching
    outN = rxLLR;
    syndromeBits = qF+1; % syndrome bits are the full set of frozen
 indices (change from zero-indexing)
```

```
    type = 'noRM';
elseif E > N % repetition
    outN = rxLLR(1:N);
    syndromeBits = qF+1; % syndrome bits are the full set of frozen
indices (change from zero-indexing)
    type = 'rep';
elseif K/E <= 7/16 % puncturing
    outN = zeros(N,1); % place zeros in punctured indices
    outN(end-E+1:end) = rxLLR;
    syndromeBits = setdiff(qF,qFPunc)+1; % syndrome bits don't include
punctured indices
    type = 'punc';
else % shortening
    outN = 9e20*ones(N,1);  % place large value in shortened indices
    outN(1:E) = rxLLR;
    syndromeBits = setdiff(qF,qFtmp)+1; % syndrome bits don't include
shortened indices
    type = 'short';
end
```

Sub-block deleaving

```
out = zeros(N,1); % initialize out
out(jn+1) = outN; % perform deleaving
```

Map LLR values to bits

```
out(out >= 0) = 0;
out(out < 0) = 1;
```

Modified polar decoding

```
% Get G, nth Kronecker power of kernel
n = log2(N);
ak0 = [1 0; 1 1];   % Arikan's kernel
allG = cell(n,1);   % Initialize cells
for i = 1:n
    allG{i} = zeros(2^i,2^i);
end
allG{1} = ak0;      % Assign cells
for i = 1:n-1
    allG{i+1} = kron(allG{i},ak0);
end
G = allG{n};
```

```
% Decode using matrix multiplication
z = mod(out'*G,2)';
```

Modified Syndrome Query

```
% check if syndrome table exists in working directory, if not create it
if isfile(sprintf('K%d_E%d_N%d.mat',K,E,N))
    load(sprintf('K%d_E%d_N%d.mat',K,E,N))
else
        % Ensure the function is saved to the appropriate file name
before running
    if E >= N % for no rate-match and repetition, the syndrome table can
be computed directly
        s = zeros(N,2^15);
        syndromeTable = zeros(N,2^15);
        for n = 1:2^15
            s(:,n) = nrPDCCHPRBS(nID,n,N); % determine scrambling
sequence
            s(jn+1,n) = s(:,n); % deleave
            syndromeTable(:,n) = mod(s(:,n)'*G,2); % polar decoding
        end
        filename = sprintf('K%d_E%d_N%d.mat',K,E,N);
        save(filename,'syndromeTable')

    else % for puncturing and shortening, full process must be gone
through using dummy data segment and rnti value

        nRNTID = randi([1 65519]);
        aD = randi([0 1],[K-24,1]);

        cVecD = nrCRCEncode(aD,"24C",nRNTID);

        % Input is a single code block and assumes CRC bits are included
        piD = nr5g.internal.polar.interleaveMap(K);
        inIntrD = cVecD(piD+1);

        % Get sequence for N, ascending ordered
        s10D = nr5g.internal.polar.sequence;          % Nmax=10
        idxD = (s10D < N);
        qSeqD = s10D(idxD);                            % 0-based

        % Get frozen, information bit indices sets, qF, qI
        jn = nr5g.internal.polar.subblockInterleaveMap(N);  % 0-based
        qFtmp = [];
        if E < N
            if K/E <= 7/16  % puncturing
                for i = 0:(N-E-1)
                    qFtmp = [qFtmp; jn(i+1)];   % punctured bits
                end
                if E >= 3*N/4 % extra freezing
                    uLim = ceil(3*N/4-E/2);
                    qFtmp = [qFtmp; (0:uLim-1).'];
                else % extra freezing
                    uLim = ceil(9*N/16-E/4);
                    qFtmp = [qFtmp; (0:uLim-1).'];
                end
                qFtmp = unique(qFtmp);
            else            % shortening
                for i = E:N-1
                    qFtmp = [qFtmp; jn(i+1)];   %#ok
                end
            end
        end

        % Get qI from qFtmp and qSeq
        qI = zeros(K,1);
        j = 0;
        for i = 1:N
            ind = qSeq(N-i+1);      % flip for most reliable
            if any(ind==qFtmp)
                continue;
            end
            j = j+1;
            qI(j) = ind;
            if j==(K)
                break;
            end
        end

        % Form the frozen bit vector
        qF = setdiff(qSeq,qI);     % sorted doesn't matter now

        F = zeros(N,1);
        F(qF+1) = ones(length(qF),1);

        % Generate u
        uD = zeros(N,1);     % doubles only

        % CRC-Aided Polar (CA-Polar)
        uD(F==0) = inIntrD;   % Set information bits (interleaved)
```

```
        % Encode using matrix multiplication
        encOutD = mod(uD'*G,2)';

        % Sub-block interleaving
        yD = encOutD(jn+1);

        % Bit selection
        if K/E <= 7/16
            % puncturing (take from the end)
            outED = yD(end-E+1:end);
        else
            % shortening (take from the start)
            outED = yD(1:E);
        end

        cSeqD = nrPDCCHPRBS(nID,nRNTID,length(outED));
        scrambledD = xor(outED,cSeqD);

        symD = nrSymbolModulate(scrambledD,'QPSK');

        rxScrLLRD = nrSymbolDemodulate(symD,"QPSK");

        rxLLRD = NaN(E,2^15+1);
        rxLLRD(:,end) = rxScrLLRD(:,1);

        for n = 1:2^15
            descrambleSequenceD =
nrPDCCHPRBS(nID,n,E,"MappingType","signed");
            rxLLRD(:,n) = rxScrLLRD.*descrambleSequenceD;
        end

        if K/E <= 7/16
            % puncturing (put at the end)
            outND = zeros(N,2^15+1);          % 0s for punctures
            for n = 1:2^15+1
                outND(end-E+1:end,n) = rxLLRD(:,n);
            end
        else
            % shortening (put at the start)
            outND = 1e20*ones(N,2^15+1);      % use a large value for 0s
            for n = 1:2^15+1
                outND(1:E,n) = rxLLRD(:,n);
            end
        end
        % Sub-block deinterleaving
        outD = zeros(N,2^15+1);
        outD(jn+1,:) = outND;

        outD(outD >= 0) = 0;
        outD(outD < 0) = 1;

        control = mod(outD(:,end)'*G,2)';
        syndromeTable = NaN(N,2^15);

        for n = 1:2^15
            temp = mod(outD(:,n)'*G,2)';
            syndromeTable(:,n) = xor(temp,control);
        end

        filename = sprintf('K%d_E%d_N%d.mat',K,E,N);
        save(filename,'syndromeTable')
    end
end
```

Determine matching syndromes

```
% define determination threshold

for n = 1:2^15 % for each syndrome in syndrome table...

    %%% *** Comment out the next six lines if using Hamming distance
    qFcheck = ~xor(z(syndromeBits),syndromeTable(syndromeBits,n))'; %
compare error pattern to given syndrome
    bitChange = find(diff([0,qFcheck,0]==1)); % find differences between
error pattern syndrome table
    startIndex = bitChange(1:2:end-1); % determine starting indices of
matching strings
    bitCount = bitChange(2:2:end)-startIndex; % determine lengths of
matching strings
    lcson=1;
    if max(bitCount) >= threshold

    %%% *** Comment out the next three lines if using LCS method.
     %ham = pdist([z(syndromeBits)';
syndromeTable(syndromeBits,n)'],"hamming")*length(syndromeBits);
     %lcson=0;
     %if ham <= threshold
        threshold_matches=threshold_matches+1;
```

```
         scrSeq = nrPDCCHPRBS(nID,n,E,"MappingType","signed"); % determine
signed scrambling sequence
         decodeIntLLR = rxLLR.*scrSeq; % apply scrambling sequence

         switch type
             case 'noRM' % no rate matching
                 decodeIntLLR_RM = decodeIntLLR;
             case 'rep' % repetition
                 decodeIntLLR_RM_temp = zeros(N,2);
                 decodeIntLLR_RM_temp(1:N,1) = decodeIntLLR(1:N);
                 decodeIntLLR_RM_temp(1:E-N,2) = decodeIntLLR(N+1:E);
                 decodeIntLLR_RM = sum(decodeIntLLR_RM_temp,2); % sum
repeated values
             case 'punc' % puncturing
                 decodeIntLLR_RM = zeros(N,1);
                 decodeIntLLR_RM(end-E+1:end) = decodeIntLLR; % place
zeros at start
             case 'short' % shortening
                 decodeIntLLR_RM = 9e20*ones(N,1);
                 decodeIntLLR_RM(1:E) = decodeIntLLR; % place large values
at end
         end

         decodeLLR = zeros(N,1); % initialize decodeLLR
         decodeLLR(jn+1) = decodeIntLLR_RM; % deleave

         dataIntScr=nrPolarDecode(decodeLLR,K,E,1,9,logical(0),24);
%#ok<LOGL> %polar decoding

         pi = nr5g.internal.polar.interleaveMap(K);  % obtain deleaving
indicies
         dataScr = NaN(1,length(pi)); % initialize dataScr
         dataScr(pi+1) = dataIntScr; % deleave data + crc bits
         [data, CRCerr]=nrCRCDecode([ones(24,1); dataScr'],'24C',n); %pad
with 24 ones as nrDCIEncode calculates CRC with this padding

         %need to determine  how to capture n+2^15 case - must do valid
bit
         if CRCerr==0 % if all crc bits check, rnti is n
             detRNTI=n;
             break
         elseif CRCerr==2^15 % if all but 1 crc bits check, rnti is n+2^15
             detRNTI=n+2^15;
             break
         end
```

```
     end %end loop checking for syndromes that meet threshold

     if n == 2^15 % if syndrome table searched through without finding
match, give no result message
         detRNTI=-1;
     end
 end %end search threshold for each possible syndrome

 %update total number of matches, errors
 if detRNTI==-1
     nomatch=nomatch+1;
 elseif detRNTI==nRNTI
    match=match+1;
 else
     false=false+1;
 end

 end %end tests loop
 toc

 %track values and log at threshtotalcounter which continually increments
 tracker(threshtotalcounter,1)=payload;      %log current payload
 tracker(threshtotalcounter,2)=threshold;   %log current threshold
 tracker(threshtotalcounter,3)=match;        %out of number of tests
 tracker(threshtotalcounter,4)=nomatch;      %out of number of tests
 tracker(threshtotalcounter,5)=false;        %out of number of tests
 tracker(threshtotalcounter,6)=threshold_matches;    %log total
threshold_matches for thresh/payload combo

 %sum values from tracker and hold in long term logs
 type1errors(payloadcounter,threshcounter)=(tracker(threshtotalcounter,6)-
tracker(threshtotalcounter,3))/tests/2^15;  %threshold matches per test
(minus correct) divided by total potential matches
 successes(payloadcounter,threshcounter)=tracker(threshtotalcounter,3)/tes
ts;            %avg total successes logged
 RNTIsmatchedavg(payloadcounter,threshcounter)=tracker(threshtotalcounter,
6)/tests;      %avg per combo
 threshcounter=threshcounter+1 %increments each threshold change, then
resets
 threshtotalcounter=threshtotalcounter+1; %continually increments each
threshold change

 %reset counters
 match=0; nomatch=0; false=0; threshold_matches=0;
 end %end threshold loop
```

```
threshcounter=threshcounter-1; %reset for last value
payloadcounter=payloadcounter+1
end %end payload loop
```

Generate Various Plots for Analysis

```
%plots
%Plot 1 Probability of Success versus Threshold
figure()
hold off
plot(thresholdvalues, successes', '-o')
ax = gca;
ax.LineStyleOrder = '-';
ax.YColor = 'black';
colororder('default')
ylabel('Probability of Success, {\it P_{success}}')
if lcson==1
    xlabel('LCS Threshold')
else
    xlabel('Hamming Threshold, {\it \tau_{HAM}}')
end
grid on
legend('{\itA = }12', '{\itA = }22', '{\itA = }32', '{\itA = }42', '{\itA
= }52', '{\itA = }63', '{\itA = }73', '{\itA = }83', '{\itA = }93', '{\itA
= }103')
legend('Location', 'northeastoutside')
figure(1)
getpdf=gcf;
exportgraphics(getpdf,'plotsuccess.pdf')

%Plot 2 Probability of Error versus Threshold
figure()
hold off
plot(thresholdvalues, type1errors','--o')
ax = gca;
ax.LineStyleOrder = '--';
ax.YColor = 'black';
colororder('default')
ylabel('Probability of Error, {\itP_{error}}')
legend('{\itA = }12', '{\itA = }22', '{\itA = }32', '{\itA = }42', '{\itA
= }52', '{\itA = }63', '{\itA = }73', '{\itA = }83', '{\itA = }93', '{\itA
= }103')
if lcson==1
    xlabel('LCS Threshold')
else
```

```
    xlabel('Hamming Threshold, {\it\tau_{HAM}}')
end
grid on
legend('Location', 'northeastoutside')
getpdf=gcf;
exportgraphics(getpdf,'ploterror.pdf')

%Plot 3 Probability of Error versus Probability of Success (log-log)
figure()
loglog(type1errors', successes','-o')
ax = gca;
ax.LineStyleOrder = '-';
ax.YColor = 'black';
colororder('default')
hold on
ylabel('Probability of Success, {\itP_{success}}')
legend('{\itA = }12', '{\itA = }22', '{\itA = }32', '{\itA = }42', '{\itA
= }52', '{\itA = }63', '{\itA = }73', '{\itA = }83', '{\itA = }93', '{\itA
= }103')
xlabel('Probability of Error, {\itP_{error}}')
grid on
legend('Location', 'northeastoutside')
getpdf=gcf;
exportgraphics(getpdf,'plotmixed.pdf')
axis([10E-9 1 10E-3 1])

%Plot 4 Probability of Success versus Number of Matched RNTIs
figure()
hold off
for plotcount=1:(length(payloadlengths))
    plot(RNTIsmatchedavg(plotcount,:), successes(plotcount,:),'-o')
    hold on
end
ax = gca;
ax.LineStyleOrder = '--';
ax.YColor = 'black';
colororder('default')
ylabel('Probability of Success, {\itP_{success}}')
legend('{\itA = }12', '{\itA = }22', '{\itA = }32', '{\itA = }42', '{\itA
= }52', '{\itA = }63', '{\itA = }73', '{\itA = }83', '{\itA = }93', '{\itA
= }103')
xlabel('Matched RNTIs')
grid on
legend('Location', 'northeastoutside')
```

```
%Plot 5 Probability of Error versus Threshold (semilog)
figure()
hold off
semilogy(thresholdvalues, type1errors','--o')
ax = gca;
ax.LineStyleOrder = '--';
ax.YColor = 'black';
colororder('default')
ylabel('Probability of Error, {\it P_{error}}')
legend('{\itA = }12', '{\itA = }22', '{\itA = }32', '{\itA = }42', '{\itA
= }52', '{\itA = }63', '{\itA = }73', '{\itA = }83', '{\itA = }93', '{\itA
= }103')
axis([0 100 10e-9 1])
if lcson==1
    xlabel('LCS Threshold')
else
    xlabel('Hamming Threshold, {\it \tau_{HAM}}')
end
grid on
legend('Location', 'northeastoutside')

%Plot 6 Probability of Success versus Matched RNTIs (semilog)
figure()
hold off
for plotcount=1:(length(payloadlengths))
    semilogx(RNTIsmatchedavg(plotcount,:), successes(plotcount,:),'-o')
    hold on
end
ax = gca;
ax.LineStyleOrder = '--';
ax.YColor = 'black';
colororder('default')
ylabel('Success Rate')
legend('{\itA = }12', '{\itA = }22', '{\itA = }32', '{\itA = }42', '{\itA
= }52', '{\itA = }63', '{\itA = }73', '{\itA = }83', '{\itA = }93', '{\itA
= }103')
xlabel('Matched RNTIs')
grid on
legend('Location', 'northeastoutside')

%Plot 7 Probability of Success versus Probability of Error
figure()
plot(type1errors', successes','-o')
ax = gca;
```

```
 ax.LineStyleOrder = '-';
 ax.YColor = 'black';
 colororder('default')
 hold on
 ylabel('Probability of Success, {\itP_{success}}')
 legend('{\itA = }12', '{\itA = }22', '{\itA = }32', '{\itA = }42', '{\itA
= }52', '{\itA = }63', '{\itA = }73', '{\itA = }83', '{\itA = }93', '{\itA
= }103')
 xlabel('Probability of Error, {\itP_{error}}')
 grid on
 legend('Location', 'northeastoutside')
 getpdf=gcf;
 exportgraphics(getpdf,'plotmixed.pdf')

 %Plot 8 3D Plot of PRobability of Success versus Probability of Error and
 %Threshold
 figure()
 plot3(thresholdvalues, type1errors', successes')
 grid on
 xlabel('Hamming Threshold, {\it\tau_{HAM}}')
 zlabel('Probabililty of Success')
 ylabel('Probability of Error')
 legend(strsplit(num2str((payloadlengths))))

 %Plot (1X2) Probability of Success verus Threshold and Proabability of
 %Error
 figure()
 subplot(1,2,1)
 box on
 hold on
 plot(thresholdvalues, successes', '-o')
 ax = gca;
 ax.LineStyleOrder = '-';
 ax.YColor = 'black';
 colororder('default')
 ylabel('Probability of Success, {\itP_{success}}')
 legend('{\itA = }12', '{\itA = }22', '{\itA = }32', '{\itA = }42', '{\itA
= }52', '{\itA = }63', '{\itA = }73', '{\itA = }83', '{\itA = }93', '{\itA
= }103')
 legend('Location', 'northwest')
 axis([0 100 0 1])
 ylabel('Probability of Success, {\itP_{success}}')
 if lcson==1
     xlabel('LCS Threshold')
 else
```

```
    xlabel('Hamming Threshold, {\it\tau_{HAM}}')
end

grid on
subplot(1,2,2)
hold off
plot(thresholdvalues, type1errors','--o')
ax = gca;
ax.LineStyleOrder = '--';
ax.YColor = 'black';
colororder('default')
ylabel('Probability of Error, {\itP_{error}}')
legend('{\itA = }12', '{\itA = }22', '{\itA = }32', '{\itA = }42', '{\itA
= }52', '{\itA = }63', '{\itA = }73', '{\itA = }83', '{\itA = }93', '{\itA
= }103')
if lcson==1
    xlabel('LCS Threshold')
else
    xlabel('Hamming Threshold, {\it\tau_{HAM}}')
end
grid on
legend('Location', 'northwest')
endmodel=toc(startmodel)
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Ericsson, “Ericsson Mobility Report, June 2022”, Stockholm, Sweden, 2022 [Online]. Available: https://www.ericsson.com/49d3a0/assets/local/reports-papers/mobility-report/documents/2022/ericsson-mobility-report-june-2022.pdf

[2] *Security architecture and procedures for 5G System*, 3GPP TS 33.501 version 15.4.0 Release 15, 2019 [Online]. Available: https://www.etsi.org/deliver/etsi_ts/133500_133599/133501/15.04.00_60/ts_133501v150400p.pdf

[3] B. Gardner, “An efficient methodology to de-anonymize the 5G-New Radio Physical Downlink Control Channel”, M.S. thesis, Dept. of Electrical and Computer Engineering, NPS, Monterey, CA, US, 2020 [Online]. Available: https://calhoun.nps.edu/handle/10945/65524

[4] P. Chandra, Bulletproof Wireless Security, GSM, UMTS, 802.11 and Ad Hoc Security, Burlington, MA, USA: Newnes, 2005.

[5] Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; 3GPP System Architecture Evolution (SAE); Security Architecture, 3GPP TS 33.401 version 15.7.0 Release 15, 2019 [Online]. Available: https://www.etsi.org/deliver/etsi_ts/133400_133499/133401/15.07.00_60/ts_133401v150700p.pdf]

[6] S. Rommer, P. Hedman, M. Olsson, L. Frid, S. Sultana, and C. Mulligan, *5G Core Networks, Powering Digitalization*, San Diego, CA, USA: Academic Press, 2020.

[7] Qualcomm, “3GPP Release 17: Completing the first phase of the 5G evolution”, San Diego, CA, 2022, [Online]. Available: https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/download-our-5g-nr-rel-17-presentation.pdf

[8] *NR; NR and NG-RAN Overall Description; Stage 2 (Release 15)*, 3GPP Standard 38.300, 2021 [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3191

[9] *NR; Multiplexing and channel coding (Release 15)*, 3GPP Standard 38.212, 2021 [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3214

[10] Z. B. K. Egilmez, L. Xiang, R. G. Maunder, and L. Hanzo, “The Development, Operation and Performance of the 5G Polar Codes*,” IEEE Communications surveys and tutorials*, vol. 22, no. 1, pp. 96-122, 2020 [Online]. Available: https://doi.org/10.1109/COMST.2019.2960746

[11] J. Garrett, “An evaluation of de-anonymization attacks against physical downlink shared channel data in 5G new radio”, M.S. thesis, Dept. of Electrical and Computer Engineering, NPS, Monterey, CA, US, 2021 [Online]. Available: https://calhoun.nps.edu/handle/10945/68323

[12] W. Stallings, *Data and Computer Communications*, 9th ed. Upper Saddle River, NJ, USA: Pearson, 2011.

[13] T. T. Ha, *Theory and Design of Digital Communication Systems*, New York, NY, USA: Cambridge University Press, 2011.

[14] W. Stallings, *5G Wireless: A Comprehensive Introduction*, Boston, MA, USA: Addison-Wesley, 2021.

[15] I. Taland and A. Vardy, “List decoding of polar codes,” *IEEE Trans. Inf. Theory*, vol.61, pp. 2213–2226, May2015.

[16] K. Niuand and K. Chen, “CRC-aided decoding of polar codes,” *IEEE Commun. Lett.*, vol.16, pp. 1668–1671, October 2012.

[17] C. Johnson, *5G New Radio in Bullets*. Farnham, England: Chris Johnson, 2019.

[18] *NR; Physical channels and modulation (Release 15)*, 3GPP Standard 38.211, 2021 [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3213

[19] *NR; Medium Access Control (MAC) protocol specification (Release 15)*, 3GPP Standard 38.321, 2021 [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3194

[20] L. Xiang, Z. B. K. Egilmez, R. G. Maunder, and L. Hanzo, “CRC-Aided Logarithmic Stack Decoding of Polar Codes for Ultra Reliable Low Latency Communication in 3GPP New Radio,” *IEEE access*, vol. 7, pp. 28559-28573, 2019 [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2901596

[21] *NR; Physical layer procedures for control (Release 15)*, 3GPP Standard 38.213, 2021 [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3215

[22] K. Takeda, H. Xu, T. Kim, K. Schober and X. Lin, "Understanding the Heart of the 5G Air Interface: An Overview of Physical Downlink Control Channel for 5G New Radio," *IEEE Communications Standards Magazine*, vol. 4, no. 3, pp. 22-29, September 2020 [Online]. Available: https://doi.org/10.1109/MCOMSTD.001.1900048

[23] V. Chvatal and D. Sankoff, "Longest common subsequences of two random sequences," *Journal of applied probability*, vol. 12, no. 2, pp. 306-315, 1975 [Online]. Available: https://doi.org/10.2307/3212444

[24] S. N. Majumdar and S. Nechaev, "Exact asymptotic results for the Bernoulli matching model of sequence alignment," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol 72, no. 2 Pt 1, pp. 020901-020901, 205 [Online]. Available: https://doi.org/10.1103/PhysRevE.72.020901

[25] A. Schacht, "Position Estimate Fidelity from TAG Multilateration Attack within the 5G Environment", M.S. thesis, Dept. of Electrical and Computer Engineering, NPS, Monterey, CA, US, 2021 [Online]. Available: https://calhoun.nps.edu/handle/10945/67177

[26] K. Foster, "Implications for Location Privacy in 5G", M.S. thesis, Dept. of Electrical and Computer Engineering, NPS, Monterey, CA, US, 2021 [Online]. Available: https://calhoun.nps.edu/handle/10945/67712

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California