Jamming Detection And Mitigation in Spread Spectrum Communication

K.Saranya¹, B.Kharthika²

¹PG student, Department of ECE, RVS Educational Trust’s and Group of Institutions, Dindigul, TN
²Assistant Professor, Department of ECE, RVS Educational Trust’s and Group of Institutions, Dindigul, TN

Email: saranya.krish987@gmail.com

Abstract —
Some secret is shared between the communication nodes before establishment the spreading sequence in spread spectrum (SS) wireless communication. This pre-configuration of secrets causes the physical contact as infeasible in the large networks in which the nodes are join and leave the network frequently and it is called Circular Dependency Problem (CDP). To solve CDP, introduce an efficient and adversary-resilient secret sharing mechanism. This method uses MAC-Masked Key Scheduling. This key scheduling fails to protect against Brute-Force jammers. So, we introduce a key scheduling with linear tail. Communication nodes enter and leave the networks dynamically, making pre-configuration impractical. To achieve an efficient spread spectrum communication can avoid the adversary effects like Jamming, Reply attack, Modification, Insertion. These kinds of jammer activities cause the failure in establishment of communication. To protect this kind of attacks from adversary, we introduce key scheduling. This paper deals about how to protect against such kind of adversary activities.

Key words—Spread spectrum, CDP, Key scheduling mechanism, MAC, Linear tail.

1. INTRODUCTION

Wireless technology is becoming more and more popular and is widely used by companies and individuals for important communications, such as mobile e-commerce transactions, email, and corporate data transmissions. As the result, security issues become more and more important for wireless networks. This is not a trivial problem because wireless devices, including smart cellular phones and personal digital assistants (PDAs) with Internet access, were not originally designed with security as a top priority. Most of wireless network security problems can be mitigated or fully addressed by changing wireless network
security architectures or using more advanced cryptographic methods. However, there are still some threats that cannot be addressed by these methods.

Jamming is an important class of such threats. Due to the openness of the wireless medium, attackers can easily implement jamming attacks to inject signals into the medium. Attackers can easily observe communications between legitimate users, and then make the transmission in wireless networks fail by injecting false messages. The attackers can implement different kinds of jamming attacks.

1. Constant jammer: The constant jammer continually emits a radio signal.

2. Deceptive jammer: The deceptive jammer constantly injects regular packets to the channel. So legitimate users will be deceived into believing the jammer is also a legitimate user in transmitting state.

3. Random jammer: The random jammer alternates between sleeping and jamming.

4. Reactive jammer: The reactive jammer only begins jamming when the jamming detects activity in the channel.

Reactive jamming is the most important threat among the four jammers. The reason is that, while destroying the packets, the attacker minimizes its risk of being detected. In frequency hopping, reactive jammer cannot complete the detection process if the hopping rate is high enough. So it is can be seen that to mitigate jamming, the spread spectrum techniques are usually adopted.

Spread spectrum techniques are conventional anti-jamming methods. The spread spectrum signals usually have the characteristic that the bandwidth is much larger than the information rate which can be seen as redundancy. This kind of redundancy is added to the signal due to the signal is required to overcome severe interference in the process of transmission in the channel. The redundancy of the spread spectrum signal can be characterize by bandwidth expansion factor which is usually much larger than one. To introduce redundancy to signal, we know that coding is an efficient method. So how to code the signal to make it spread spectrum is the first key element in designing the spread spectrum systems.

In the security aspect, in order to avoid the attacker to demodulate the spread spectrum signals, pseudorandomness is needed. The pseudorandomness of the spread spectrum 3 signals makes the signals seem to be random noise to the attacker thus making it very difficult for the attacker to demodulate the signals. This characteristic is actually related to the purpose or application of these spread spectrum signals. In, the authors list the main purposes of the spread spectrum signals:
1. To combat the effects of interference due to jamming, interference caused by other users of the channel and self-interference due to multipath propagation.

2. To hide a signal by transmitting it at low power, thus making it difficult for an attacker to detect the signal in the presence of background noise.

3. To achieve message privacy in the presence of attackers.

4. To obtain accurate range (time delay) and range rate (velocity) measurements in radar and navigation (this purpose is not directly related to communications). In combating the effects of interference of intentional jamming, the knowledge of the jammer is important. If the jammer knows the characteristic of the transmitting signal, it is easy for the jammer to mimic this signal transmitted by the transmitter and confuse the receiver. To prevent this to happen, the transmitter introduces the randomness (actually pseudorandomness) to the signal which is unpredictable for the jammer while known to the receiver. So the only way for the jammer to do jamming is to transmit an interfering signal without any prior knowledge about the pseudorandom pattern.

Frequency-hopping spread spectrum (FHSS), direct-sequence spread spectrum (DSSS), time-hopping spread spectrum (THSS), chirp spread spectrum (CSS), and combinations of these techniques are forms of spread spectrum. Each of these techniques employs pseudorandom number sequences created using pseudorandom number generators to determine and control the spreading pattern of the signal across the allocated bandwidth.

The traditional frequency hopping (FH) which relies on secrets shared between the transmitter and receiver. The shared secret then determines the hopping pattern in FH. A third party who does not know the secret codes cannot predict the transmission. Then the probability of the transmission being jammed is reduced. But the prerequisite is a secret must be shared by the transmitter and the receiver.

2. RELATED WORK

In cryptography, a pre-shared key or PSK is a shared secret which was previously shared between the two parties using some secure channel before it needs to be used. To build a key from shared secret, the key derivation function should be used. Such systems almost always use symmetric key cryptographic algorithms. The term PSK is used in Wi-Fi encryption such as WEP or WPA, where both the wireless access points (AP) and all clients share the same key.

The characteristics of this secret or key are determined by the system which uses it; some system designs require that such keys be in a particular format. It can be a password, a passphrase, or a hexadecimal string. The secret is used by all systems involved in the cryptographic processes used to secure the traffic between the systems. Our system does not pre sharing of secret key is called zero preshared key. Unlike conventional DSSS, the key K is not known to anyone but S when transmission starts. Both the length of message l and
key length k are public information. The bits in M are 0 or 1 with equal probability. If they are not, they can be compressed. The details about our scheme, including the underlying algorithm for PN generation, are public information.

Some secret is shared between the communication nodes before the establishment the spreading sequence in spread spectrum (SS) wireless communication. This pre configuration of secrets causes the physical contact as infeasible in the large networks in which the nodes are join and leave the network frequently and it is called Circular Dependency Problem (CDP). Communications System Toolbox™ provides algorithms for designing, simulating, and analyzing communications systems.

These capabilities are provided as MATLAB® functions, MATLAB System objects™, and Simulink® blocks. The system toolbox enables source coding, channel coding, interleaving, modulation, equalization, synchronization, and channel modeling. You can also analyze bit error rates, generate eye and constellation diagrams, and visualize channel characteristics. Using adaptive algorithms, you can model dynamic communications systems that use OFDM, OFDMA, and MIMO techniques. Algorithms support fixed-point data arithmetic and C or HDL code generation.

- Algorithms for designing the physical layer of communications systems, including source coding, channel coding, interleaving, modulation, channel models, MIMO, equalization, and synchronization
- GPU-enabled System objects for computationally intensive algorithms such as Turbo, LDPC, and Viterbi decoders.
- Eye Diagram Scope app and visualization functions for constellations and channel scattering.
- Bit Error Rate app for comparing the simulated bit error rate of a system with analytical results.
- Channel models, including AWGN, Multipath Rayleigh Fading, Rician Fading, MIMO Multipath Fading, and LTE MIMO Multipath Fading.
- Basic RF impairments, including nonlinearity, phase noise, thermal noise, and phase and frequency offsets.
- Algorithms available as MATLAB functions, MATLAB System objects, and Simulink blocks.
- Support for fixed-point modeling and C and HDL code generation.

2. SYSTEM MODEL

![Diagram](image)

Fig.1. System model
System model(fig.1) shows that the Sender Sends efficiently the message to the Destination without the jammer effect to establish the communication the sender uses the cryptographically derived PN sequence, Receiver MAC address and Key.

This system based on two mechanism namely, Intractable forward decoding, Efficient Backward decoding. This mechanism uses Time Reversed Message Extraction and Key Scheduling (TREKS). To avoid the jamming it uses Zero preshared key DSSS and MAC-Masked Key Scheduling. Intractable forward decoding enhances the time required to find the key is higher than the packet transmission time. Efficient backward decoding used for the key inferring and message despreading.

Then the proposed system uses the key scheduling with linear tail. If the message transmission time is less than or equal to radio turn-around time that leads for jamming. So, we make last 10 bits of the messages are in the linear rate of 1 key per packet bit and first ten message segments are in reduced size. The protection against brute force jammers is achieved by choosing the different message partition and entropy reduction value. It solves circular dependency problem.

3.1. Modules

- **PN sequence and Message**: Sender(S) randomly chooses a secret key K of length k bits. Sender then uses K to generate a cryptographically strong PN-sequence to spread M. Both the length of message (l) and key length k are public information. The bits in Message(M) are 0 or 1 with equal probability. If they are not, they can be compressed. Algorithm for PN generation is also public information.

- **Message spreading**: To create jam the enemy wants to know about the key. The enemy chooses the key (K) and key length (k) at uniformly random. Intractable forward decoding creates the time required to find the key is higher than the packet transmission time, then the jammer miss the chance to jam the transmission. The message is transmitted before the jammer finding the key. Then we spread the message with a PN sequence derived cryptographically.

- **Key scheduling**: We use the key scheduling with linear tail. In this key scheduling we avoid jamming by creating the message transmission time is less than the radio turn-around time of the jammer. We also make last 10 bits of the message at a linear rate of one key bit per packet bit and we also reduce the entropy rate. This key scheduling protect against brute force jammers. By using this key scheduling the total size of the packet is reduced.

- **Efficient backward decoding**: Efficient backward decoding reduces the computational complexity. The receiver can deduce the key due to the decreasing key entropy by guessing two keys to find the end of transmission. The receiver can store the received signal and then process this signals as
backwards in time. Time Reversed Message Extraction consists of 2 phases namely, Finding the EOM, Key inferring and Message Extraction.

- **Sampling and buffering**: To find EoM and achieve bit synchronization, the receiver computes the correlation between the signal and the expected PN sequence. Because of its computational expensive, we use Fast Fourier Transform (FFT). FFT also reduces the cost. After finding the EoM, the receiver only keep 2nl chips in his buffer because he will have to traverse at most nl chips before he recovers the message. All vectors exceeding a given correlation are transferred to the next phase for further processing.

- **Key inferring and despreading of message**: We use the time reversed bitwise key inferring. In this method we try two possibilities for the current key bit. We count the number of peaks seen in the segment. If the number of peaks is greater than half the number expected bits of the segment, we assume the value at the corresponding key bit position to be correct and move onto the next. Otherwise, we abort the key inferring, implying a packet loss. Once the key has been inferred, we despread the message.

### 3.2. Data Encryption Standard

It is a previously predominant symmetric-key algorithm for the encryption of electronic data. DES is the archetypal block cipher - an algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations into another cipher text bit string of the same length. In the case of DES, the block size is 64 bits. DES also uses a key to customize the transformation, so that decryption can supposedly only be performed by those who know the particular key used to encrypt. The key ostensibly consists of 64 bits; however, only 56 of these are actually used by the algorithm.

Eight bits are used solely for checking parity, and are thereafter discarded. Hence the effective key length is 56 bits, and it is always quoted as such. The key is nominally stored or transmitted as 8 bytes, each with odd parity. One bit in each 8-bit byte of the KEY may be utilized for error detection in key generation, distribution, and storage. Bits 8, 16, ..., 64 are for use in ensuring that each byte is of odd parity. Like other block ciphers, DES by itself is not a secure means of encryption but must instead be used in a mode of operation. We use DES algorithm to generate the Pseudorandom sequence cryptographically.

### 3.3. Jamming Detection Algorithm

When the transmitter sends a packet, it will do so on the minimal disjoint cover on which no jamming had been previously detected, so that all legitimate receivers can decode the packet. In order to detect additional jammers, the transmitter additionally transmits on a test hopping pattern, which it randomly chooses from among the descendants of the cover.
This redundant test hopping pattern allows the transmitter and receiver to cooperatively detect jamming on any hopping pattern in the cover that is an ancestor of the test hopping pattern. We call this ancestor the detectable hopping pattern. If no jammers are present, each user should get either one or two identical messages, the first encoded using one of the patterns from the cover, and possibly a second encoded using the test hopping pattern. If any user receives the second message without receiving the first message, then it should suspect jamming on the detectable hopping pattern.

Any user detecting jamming in this way should report that finding to the transmitter, for example by transmitting a Jamming Detected message using the leaf hopping pattern shared between the transmitter and the detecting receiver (because no jammer knows that leaf hopping pattern). In some instances, jamming on the detectable hopping pattern will not be detected. This can happen either when a jammer jams on the test hopping pattern or when no normal users know the test hopping pattern.

3.4. Response to Jamming

When a transmitter detects jamming, it will choose a different cover. In particular, if jamming is detected on some hopping pattern h in the current cover, the transmitter will remove h from the cover and add the two children of h to the cover. For security reasons, jamming reports are only accepted from hosts that should know hopping pattern h. If the message transmission time is less than or equal to radio turn-around time that leads for jamming. So, we make last 10 bits of the messages are in the linear rate of 1 key per packet bit and first ten message segments are in reduced size.

The protection against brute force jammers is achieved by choosing the different message partition and entropy reduction value. It solves circular dependency problem. In addition, we do not consider the case where the adversary can completely block the propagation of the radio signal from the sender to the receiver. If the adversary has unlimited power and continuously jams the channel with a strong signal, then it can obviously reduce the throughput to zero, just like it would on traditional SS.

![Fig.2. Block diagram of system modules](image-url)
Figure 2 consists of six blocks: PN sequence and message, intractable forward decoding, key scheduling with linear tail, key inferring, time reversed message extraction, and original message. In this project, message mix with PN sequence then it will be an unpredictable form then passed to the possible key scheduling mechanisms then finally produces the original message.

4. PERFORMANCE EVALUATION

In this section, we discuss about the simulation setup, and the hardware/software specification of experimental runs. Then, we will present the types of jammers we consider for evaluation, and finally the performance results with and without jammers.

4.1. Flow chart

![Flow chart diagram]

Fig.3. Block diagram of overall performance
This block diagram shows (Fig.3) the overall performance operation. It consists of eight blocks, four blocks for sender side operation and remaining blocks for receiver side operation.

Hardware requirement consists of,

- Pentium IV – 2.7 GHz
- 1GB DDR RAM
- 250Gb Hard Disk

Software requirement consists of,

- Operating System : Windows XP
- Tool : MATLAB
- Version : 7.9

4.2. Simulation description

MATLAB is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numerical computation. Using MATLAB, you can solve technical computing problems faster than with traditional programming languages, such as C, C++, and Fortran. MATLAB (Matrix Laboratory)

- high-performance language for technical computing
- computation, visualization, and programming in an easy-to-use environment

Typical uses include:

- Math and computation
- Algorithm development
- Modelling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including Graphical User Interface building.

Matlab is a data analysis and visualization tool which has been designed with powerful support for matrices and matrix operations. As well as this, Matlab has excellent graphics capabilities, and its own powerful programming language. One of the reasons that Matlab has become such an important tool is
through the use of sets of Matlab programs designed to support a particular task. These sets of programs are called toolboxes, and the particular toolbox of interest to us is the image processing toolbox. Rather than give a description of all of Matlab's capabilities, we shall restrict ourselves to just those aspects concerned with handling of images. We shall introduce functions, commands and techniques as required. A Matlab function is a keyword which accepts various parameters, and produces some sort of output: for example a matrix, a string, a graph. Examples of such functions are sin, imread, imclose. There are many functions in Matlab, and as we shall see, it is very easy (and sometimes necessary) to write our own.

Matlab's standard data type is the matrix all data are considered to be matrices of some sort. Images, of course, are matrices whose elements are the grey values (or possibly the RGB values) of its pixels. Single values are considered by Matlab to be matrices, while a string is merely a matrix of characters; being the string's length. In this chapter we will look at the more generic Matlab commands, and discuss images in further chapters. When you start up Matlab, you have a blank window called the Command Window in which you enter commands. Given the vast number of Matlab's functions, and the different parameters they can take, a command line style interface is in fact much more efficient than a complex sequence of pull-down menus.

You can use MATLAB in a wide range of applications, including signal and image processing, communications, control design, test and measurement financial modeling and analysis. Add-on toolboxes (collections of special-purpose MATLAB functions) extend the MATLAB environment to solve particular classes of problems in these application areas.

MATLAB provides a number of features for documenting and sharing your work. You can integrate your MATLAB code with other languages and applications, and distribute your MATLAB algorithms and applications.

MATLAB stands for Matrix Laboratory. Matlab had many functions and toolboxes to help in various applications. It allows you to solve many technical computing problems, especially those with matrix and vector formulas, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C or Fortran.

- High-level language for technical computing
- Development environment for managing code, files, and data
- Interactive tools for iterative exploration, design, and problem solving
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration
- 2-D and 3-D graphics functions for visualizing data
• Tools for building custom graphical user interfaces
• Functions for integrating MATLAB based algorithms with external applications and languages, such as C, C++, Fortran, Java, COM, and Microsoft Excel.

4.3 Interfacing with other languages

MATLAB can call functions and subroutines written in the C programming language or Fortran. A wrapper function is created allowing MATLAB data types to be passed and returned. The dynamically loadable object files created by compiling such functions are termed "MEX-files" (for MATLAB executable).

Libraries written in Perl, Java, ActiveX or .NET can be directly called from MATLAB, and many MATLAB libraries (for example XML or SQL support) are implemented as wrappers around Java or ActiveX libraries. Calling MATLAB from Java is more complicated, but can be done with a MATLAB toolbox which is sold separately by MathWorks, or using an undocumented mechanism called JMI (Java-to-MATLAB Interface), which should not be confused with the unrelated Java Metadata Interface that is also called JMI.

MATLAB has structure data types. Since all variables in MATLAB are arrays, a more adequate name is "structure array", where each element of the array has the same field names. In addition, MATLAB supports dynamic field names (field look-ups by name, field manipulations, etc.). Unfortunately, MATLAB JIT does not support MATLAB structures, therefore just a simple bundling of various variables into a structure will come at a cost.

MATLAB supports elements of lambda calculus by introducing function handles, or function references, which are implemented either in .m files or anonymous/nested functions.

MATLAB system consists of these main parts:

• Desktop Tools and Development Environment

  Includes the MATLAB desktop and Command Window, an editor and debugger, a code analyzer, browsers for viewing help, the workspace, files, and other tools

• Mathematical Function Library

  Vast collection of computational algorithms ranging from elementary functions, like sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigen values, Bessel functions, and fast Fourier transforms.

• The Language
The MATLAB language is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features.

- **Graphics**

MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as editing and printing these graphs. It also includes functions that allow you to customize the appearance of graphics as well as build complete graphical user interfaces on your MATLAB applications.

- **External Interfaces**

The external interfaces library allows you to write C and Fortran programs that interact with MATLAB.

A good choice for vision program development because:

- Easy to do very rapid prototyping
- Quick to learn, and good documentation
- A good library of image processing functions
- Excellent display capabilities
- Widely used for teaching and research in universities and industry

MATLAB consists of:

- **The MATLAB language**

  A high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features.

- **The MATLAB working environment**

  The set of tools and facilities that you work with as the MATLAB user or programmer, including tools for developing, managing, debugging, and profiling.

- **Handle Graphics**

  The MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics.

- **The MATLAB function library**
A vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms as well as special image processing related functions.

- The MATLAB Application Program Interface (API)

A library that allows you to write C and Fortran programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

4.4. Performance results

Finally we achieve the performance results using simulation software tools. Using Matlab coding to produce output screen shots. First we chooses the desired input message. Cryptographically derived PN sequence is generated for that input message. Here uses 5-segments of FFT. Each segment generate $5 \times 64$ bit for input message and also 8 bit key used for each segment which is used to reduce the jamming. But last segment assign each bit has separate key so this is completely reduce the jamming activities. This is called linear tail. This is also more efficient than the other methods.

It is possible to protect against brute-force jammers. No energy usage compared to SS with preshared keys. Efficient End of the Message (EoM) is detected. Message extracted efficiently. It does not requires receiver synchronization. Key establishment not required. This key scheduling also reduce the communication energy cost and computation and storage cost.

5. CONCLUSION

In this paper, we propose new mechanisms, design, and a full implementation of a real-time direct sequence spread spectrum system that does not require pre shared secrets between the parties. We introduce a new key scheduling, that is key scheduling with linear tail to protect against brute-force jammers. The key scheduling with linear tail involves reduced packet size. That also protect against the enemy activities like replay attack, modification and insertion. We can achieve this without establishing key. It results long term communication without communicating secrets. Finally, we evaluate both the computation cost and the achieved resiliency.
REFERENCES:

[1] Aldocassola, Tao, Jin, Guevara, Noubir, Bishal, Thapa,”


