Error correction in quantum cryptography

Mohamed Salah El Ashmawy
The American University in Cairo AUC

Thesis
2004/27

The American University in Cairo
School of Sciences and Engineering

Error Correction in Quantum Cryptography

A Thesis Submitted to

Computer Science Department

In partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

By
Mohamed Salah El Ashmawy

Under the Supervision of
Dr. Amr Goneid

April 2004

The American University in Cairo 2004/27

Error Correction in Quantum Cryptography

A Thesis Submitted by
Mohamed Salah El Ashmawy

To the department of Computer Science
April 2004

In partial fulfillment of the requirements for
the degree of Master of Science

Has been approved by

Dr. Amr Goneid
Thesis Committee Chair/ Adviser
Affiliation [redacted]

Dr. Amr El Kadi
Thesis Committee Reader/ examiner
Affiliation [redacted]

Dr. Sherif El Kassass
Thesis Committee Reader/ examiner [illegible]
Affiliation [illegible]

Dr. Mohamed Saieed Abdel-Wahab
Thesis Committee Reader/ examiner
Affiliation [redacted]

[redacted] May 30, 04 [redacted] May 31, 2004
Department Chair/ Date Dean Date
Program Director

Acknowledgement

I would like to acknowledge my supervisor Dr. Amr Goneid for the help and support he has provided me with during the course of working on this thesis. I would like to thank him for guiding me through this work and for his help especially in the mathematical modeling part. I would also like to thank the examiners Dr. Amr El Kadi and Dr. Sherif El Kassas for their efforts and for donating some of their time for reading and assessing this thesis.

I would also like to acknowledge the Computer Science Department for its efforts at maintaining and enhancing the Computer Science masters program. I would like to thank the department for its efforts at enhancing the facilities that would help the research students to work on their thesis. Moreover, I would like to thank the American University in Cairo for its support for the masters program and for its dedication to high quality learning. I would also like to thank the University for its Research Facilities such as the library.

Abstract

Quantum Cryptography is the newest branch of cryptography and it is the hottest topic now especially that quantum cryptanalysis threatens public key cryptography. Quantum cryptography does not actually encrypt the message itself but rather generates and distributes a random key between the sender and the receiver that is totally secure; this key could then be used as a key for one-time pads or Advanced Encryption Standard (AES) cryptosystems to encrypt messages between sender and receiver. It is based on a combination of the concepts of quantum physics, information theory and classic cryptographic schemes with the goal of generating a secret key (or extending a short key) between the two communicating parties. It is a promising field as it has been proven unconditionally secure against many types of attacks.

An important phase in the creation of the keys in Quantum Cryptography is the error correction phase where the two communicating parties share a preliminary key that contains some discrepancies between both parties' versions. In this phase, the two parties communicate to refine the shared key and create a shorter key that has no discrepancies.

This thesis aims at enhancing the error correction phase so that the keys generated would be created more efficiently to create longer shared keys in less time. This is achieved through the introduction of memory between the rounds of the error correction phase where a round would identify the locations it found errors in to help the next round be more focused. This would have an effect on the final shared key between the two communicating parties where the shared secret key length would increase in size. In order to be able to apply the modification, an implementation was made to the standard BB84 protocol as well as our enhanced one that uses memory between rounds. Experiments were made for both the standard and the enhanced algorithms in order to assess the effect of the enhancements that have been introduced. The superiority of the present enhanced algorithm appeared in those experiments and an evaluation parameter has increased from 2.5 to 5 when using initial key size of 2000 bits with 80 discrepancies. The evaluation parameter has increased from 4.76 to 8.6 in the case of having initial key size of 5000 bits with 100 discrepancies. The superiority of the present enhanced algorithm has been established through a mathematical model that has been formulated for the system.

Categories and Subject Descriptors
E.3. [Data]: Data Encryption

General Terms
Security

Keywords
Cryptography, Quantum Cryptography

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Chapter 1: Introduction

1.1 Introduction

The start of cryptography dates back to the times of Julius Caesar who used a cipher to keep his messages safe from the enemies (this cipher is known now as the Caesar cipher). This cipher was a simple one where the letters of the text were replaced by other letters that are three letters after them in the alphabetic order. That cipher depended on the secrecy of the encryption algorithm and if anyone knew the algorithm used to encrypt the message, he would have been able to decrypt the messages. Since then cryptography has undergone many developments till it reached the state it is in now.

Cryptography is the art of rendering a message unreadable by any unauthorized party. It refers to the study of methods for sending an encrypted message from the sender to the recipient so that only the intended recipient can decrypt it and read the message. A person who works on cryptography is called a cryptographer whereas a person who works on attempting to defeat the cryptographic methods and decrypt encrypted messages is called a cryptanalyst. Cryptanalysis is the science of recovering the plaintext (the original text of the message) from the ciphertext (the message after being encrypted) while having no privilege to do so (not knowing the encryption key). Cryptanalysis can also lead to the recovery of the key used for encryption/decryption [Molin 2001].

Cryptography is sometimes required also to do the following tasks:

1. **Authentication**: This means that the receiver could ascertain that this message is sent by the real sender and not from an intruder impersonating him [Schneier 1996].
2. **Integrity**: This means that the receiver can verify that the message has not been modified during its way from the sender to the receiver [Schneier 1996].

A cryptosystem is defined as:

"A cryptosystem is comprised of a set $\{E_e: e \in K\}$ consisting of enciphering transformations, and the corresponding set $\{D_d: d \in K\}$ of deciphering transformations. i.e. For each $e \in K$ there exists a unique $d \in K$ such that $D_d = E^{-1}e$ so that $D_d(E_e(m)) = m$

for all m $\in$ M (where M is the message to be enciphered). The keys(e,d) are called key pairs where possible e=d." [Molin 2001]

Currently, public key cryptography has been the main cryptographic system used for encryption. In particular, public key cryptography has been used to create a common key between the communicating parties. That key would be used by other encryption schemes such as Data Encryption Standard (DES) for the actual communication between the parties. However, public key cryptography is under the threat of a new kind of cryptanalysis (quantum cryptanalysis) that would be able to decrypt the messages sent between the sender and the receiver. This kind of cryptanalysis (though impractical yet) deems public key cryptography unsafe as the messages between the communicating parties is under the threat of being decrypted by any eavesdropper using that technique. Thus, there is a need for a new cryptographic system that would be safe and secure against cryptanalytic attacks. Quantum Cryptography is a cryptographic system that could satisfy this need and it has been proven to be unconditionally secure against many types of attacks [Mayers 2001].

Quantum Cryptanalysis needs quantum computers to be available to be implemented; however, quantum cryptography does not need quantum computers for implementation. Quantum cryptography is mainly a key distribution system with the aim of creating secure keys between the sender and the receiver; however, quantum computing is the whole system of using quantum effects and quantum computers to achieve needed results.

Quantum Cryptography is based on the Heisenberg uncertainty principle which states that "measuring a quantum system disturbs it and yields incomplete information about its state." Thus, eavesdropping on the quantum communication channel causes unavoidable disturbance that can be recognized by the legitimate users [Brassard 1994]. This helps in creating a system for key distribution between two people who don't initially share any secret information.

The steps for the quantum key distribution protocol are as follows:

1. Creation of a key between the two communicating parties (Alice and Bob) using one of several techniques such as polarization methods or phase methods (explained in the next sections).

2. Error Correction for the errors and discrepancies between the keys with Alice and Bob.
3. Privacy Amplification which is concerned with reduction of the eavesdropper information that might be gained during the initial creation of the key.

These steps are explained in further details in chapter 4.

1.2 Thesis Motivation

Cryptography has been dependent in its greatest part on public key cryptography during the past years. It didn't actually depend on public key cryptosystems for encrypting messages but it depended on public key cryptography to create keys that are secure from any eavesdropper and used these keys as input to symmetric cryptosystems (DES for example) to encrypt the messages that are actually sent between the sender and the receiver.

However, this whole view of cryptography is about to be changed as quantum cryptanalysis is threatening the basis of public key cryptography which is one-way functions and is threatening to be able to compute them in polynomial time on quantum computers. Quantum computers are still in their first stages and may not see the light in the near future but the threat still holds especially that quantum computers with 3 q-bits have been already developed in the labs as will be explained later in chapter 3.
This means that public key cryptography would no longer be secure in the future and there is a need for a new class of cryptography. That's why quantum cryptography has become a very hot and interesting issue in cryptography during the past few years especially that it has been proven by Dominic Mayers and many others that it is unconditionally secure [Boyer et at. 2001, Mayers 2001]. Unconditional security means that a secure result is expected to hold against all the attacks allowed by the eavesdropper assuming that he/she has unlimited processing power. Public key cryptography techniques aren't unconditionally secure [Boyer et al. 2001, Mayers 2001].

Quantum cryptography is still in the research phase and there are issues that must be solved. It is apparent that in quantum cryptography there are many issues that need to be settled on such as the communication medium, the wavelength, and the encoding. Moreover, the BB84 protocol which is the protocol used for quantum key distribution contains many phases that need a lot of enhancements and must be put on a secure basis in addition to being adequate to user requirements [Kollmitzer et al 2002] .

The number of bits in the shared key between the two communicating parties (the output of quantum cryptography and that would be used as a key between the communicating parties) depends heavily on the number of bits lost during the error correction and privacy amplification phases (phases of BB84 protocol). This protocol is the protocol for quantum key distribution and it consists of three phases which are key generation, error correction and privacy amplification. Thus, if the error correction phase keeps more bits the final shared string would become longer. It is desirable that this final shared string, which would be used as the encryption key later, be as long as possible because in this case it could be used in vernam ciphers (one time pads) to give the best possible security. Therefore, it is desirable that the error correction phase keep as much bits as possible to achieve this goal.

The error correction phase, as it's designed in the BB84 protocol, is split into a number of rounds. Each row consists of splitting the string into blocks, comparing the parities of these blocks, and discarding a bit of each block at the end. Those rounds are repeated many times till no errors are discovered in the parities for a fixed number of rounds (20 rounds are used generally). These rounds don't interact with each other (i.e. there is no memory between one round and the other) and thus they don't take advantage of each other's results [Bennet and Brassard 1984].

1.3 Thesis Objective

It's the objective of this research to make enhancements in the error correction phase of the BB84 protocol to make it more efficient and thus to decrease the number of bits lost during this phase. This would have an effect on the final shared key between the two communicating parties where the shared secret key length would increase in size. This enhancement would be made through the use of memory between the rounds of error correction where a round would identify the locations it found errors in to help the next round to be more focused. This would decrease the number of rounds used to reach the final shared string (increase the shared string size).

1.4 Thesis Outline

This thesis paper is subdivided into different sections, each containing chapters that are related to a topic. The first section contains literature review on cryptography in general. The second section is concerned with quantum cryptography, its evolution, protocols and the problems and current projects in it. The third section contains the work done in this research to achieve the thesis objective.

This first section contains literature overview on cryptography in general and quantum computing. It consists of one chapter which is chapter 2 that is concerned with the evolution of cryptography and its systems (symmetric and public key cryptography systems), the techniques for each and the drawbacks of them.

The second section contains chapters 3 and 4 and it focuses on quantum cryptography. Chapter 3 gives a quick overview of quantum computing and is concerned with the scientific basics of quantum cryptography, its history and how it is realized. Chapter 4 concentrates on the BB84 protocol which is the main protocol used for quantum cryptography and describes its details and uses an experiment done using this protocol as an illustration.

To be able to achieve the thesis objective and compare the results of using the standard BB84 protocol as described by Bennet and Brassard with the use of the newly suggested enhanced algorithm, we needed to implement both the standard algorithm and our enhanced one. The details of this implementation and the results reached during this research are described in the third section of this thesis paper.

Following is a brief outline of the steps followed in this research of the topic and a summary of the main problems encountered and resolved. The points in the outline are elaborated in the chapters of section 3 with the results.

1. Implementing the standard algorithm that was described in the BB84 protocol and was originally designed by Bennet and Brassard in 1984. The protocol was implemented and tested with the results they have reached in their experiments. [the details are in Chapter 5]

2. Implementing the proposed enhanced algorithm which contains the dismissal of a bit from being re-evaluated after it's already checked sufficient number of times. This algorithm is based on the standard algorithm but has some customizations to enhance performance and increase the size of the resultant key. [the details are in Chapter 5]

3. After implementing both algorithm (standard and enhanced), they were tested thoroughly with different values of errors, block sizes and increments in block sizes (the variables that affect the resultant key created at the end of the error correction phase). [the details are in Chapter 6]

4. After getting the results of both algorithms in step3, the results are compared for assessing the enhancement that was achieved by using the enhanced algorithm. [the details are in Chapter 6]

5. When the error rate increases above the value of 15%, the standard and enhanced algorithms give wrong results and don't calculate the error rate correctly. These erroneous results occur because the algorithm uses parity to check that 2 blocks of the key are equivalent which fails when having large number of errors in both blocks. The solution to this problem was to add sampling before entering into the algorithm. [the details are in Chapter 6]

6. A mathematical model was developed for the behavior of the standard and enhanced algorithm towards changing values of error rate, initial block size and block size increment. The results were obtained by applying both the standard and the enhanced algorithm and they are compared. [the details are in Chapter 7].

Chapter 2

Overview of Cryptography

Chapter 2: Overview of Cryptography

2.1 Introduction

Classical cryptography can be classified to symmetric key cryptosystems and Public key cryptosystems.

2.2 Symmetric Key Ciphers (Symmetric Key Cryptosystems)

A symmetric key cryptosystem is a cryptosystem where it is easy to compute the decryption key (d) knowing only the encryption key (e) (with the possibility that the encryption key is the same as the decryption key) [Schneier 1996]. These ciphers are also called secret-key algorithms and most of them require the sender and the receiver to agree on a key before they communicate which arouses the problem of how these keys are to be communicated between them if they do not meet. Moreover, the usage of the same key more than once between the sender and the receiver makes the cipher more vulnerable to attacks and may lead to the recovery of the key by a cryptanalyst that have been eavesdropping. This means that all the messages between this sender and receiver would be recovered by the cryptanalyst.

Examples of Symmetric key ciphers are

2.2.1 Substitution Ciphers

These are ciphers in which each character in the plaintext is substituted by a character in the ciphertext and the receiver inverts the substitution on the ciphertext to recover the plaintext later [Schneier 1996].

These ciphers can be divided into:

1- Monoalphabetic: Each character in the plaintext is substituted by a corresponding character in the ciphertext [Schneier 1996].

2- Homophonic: Same as Monoalphabetic but each character in the plaintext can map to one of several characters in the ciphertext [Schneier 1996].

3- Polygon: Blocks of characters are enciphered in groups (not character by character as in Monoalphabetic substitution ciphers) [Schneier 1996].

4- Polyalphabetic: This consists of **r** Monoalphabetic substitution ciphers (there are multiple Monoalphabetic substitution ciphers). For example there might be 3 substitution ciphers that are used depending on the position of the character in the plaintext [Schneier 1996].

An example of the substitution ciphers is the Caesar Cipher which is given by the equation:

$$E_e(m_j) = c_j = m_j + b \pmod n$$

Where m_j is a message block, c_j is the encryption of the message block, n is a prime number and b is 3 in Caesar cipher.

And the decryption is done using the following equation:

$$D_d(c_j) = m_j = c_j - b \pmod n$$

Simple Substitution Ciphers face the weakness of being subject to frequency analysis because they do not change the number of times that a letter appear in the plain text (and so its equivalent in the cipher text) which can be used by a cryptanalyst to recover the encryption key. Polyalphabetic Ciphers doesn't face this weakness because the frequency of the letters in the alphabet isn't preserved. This doesn't mean that Polyalphabetic Ciphers are totally safe from frequency analysis because if the variable r is known, then the ciphertext symbols can be separated into r groups and frequency analysis can be done on each group. [Schneier 1996]

2.2.2 Transposition/Permutation Ciphers

This is a symmetric-key block cryptosystem in which the keyspace is a set of permutations on {1,2..,r}. The transformation is given by

$$E_e(m) = (m_{e(1)}, m_{e(2)}, \ldots\ldots, m_{e(r)}) = c$$

And the decryption would be as follows:

$$D_d = (c_{d(1)}, c_{d(2)}, \ldots\ldots, c_{d(r)}) = m$$

Where $d = e^{-1}$

This means that this encryption just changes the places where the plain text symbols are present [Molin 2001].

2.2.3 The DES Cryptosystem

Data Encryption Standard (DES) is a symmetric key cryptosystem in which a 56-bit key is combined with the plaintext divided into blocks in a complicated way that involves permutations to produce the cipher-text blocks. It is considered one of the best and most widely used cryptosystem commercially available. However, It is believed now that DES "has reached the end of its credibility" so improvements and new algorithms are developed to increase its security such as triple DES and AES (Advanced Encryption Standard). In triple DES, DES is used to encrypt the plaintext with a key, then it is decrypted using DES by another key then encrypted again using a third key and vice versa in case of decryption [Molin 2001].

2.2.4 Vernam Cipher (One Time Pads)

This is the only perfect encryption scheme and was invented in 1917 by Gilbert Vernam at AT&T. It contains a large nonrepeating sequence of random letters that is held only by the sender and the receiver (the encryption/decryption key) and that is as long as the message to be encrypted. Each letter in this key is used to encrypt exactly one character in the plaintext (by adding to the plaintext character modulo 26) and after sending the message the sequence of letters used is destroyed and never used again [Schneier 1996].

This cryptosystem is provably secure; in fact, this is the only cryptosystem that is provably secure today and its security depends on the following conditions:

1. The key used is completely random.
2. The key is as long as the message itself. (i.e. no letter of the key is used to encrypt more than one letter in the message)
3. It is used only once then the key is changed [Kollmitzer et al. 2002].

2.2.5 Problems of Symmetric Key Cryptosystems

Most of the Symmetric key cryptosystems are vulnerable to many attacks as discussed and can't be used in critical applications that need the secure transfer of data. Even DES needs updating and will be substituted by new algorithms such as the Advanced Encryption Standard (AES) due to the increase in the computational powers of the machines and the invention of new cryptanalysis techniques that threaten to break the algorithm. It is not that these new algorithms are impossible to break, but that it would take long time (years) of computer processing time to decrypt messages today with present technology.

On the other hand, in 1996 Lov Grover discovered a quantum algorithm (based on quantum computers) to do searches more efficiently. This algorithm is speculated to improve the effectiveness of brute force attacks on DES and AES. There are no quantum computers currently available; however, there is a lot of research done on this topic. Quantum computers with 3 quantum bits are already built [Rieffel and Pollak 2000] but there are problems in scaling to increase the number of bits inside them. The power of quantum computation derives from "the exponential state spaces of multiple quantum bits" where a single quantum bit can be in a superposition of 0 and 1 and thus a register of n qbits (quantum bits) can be in a superposition of 2^n possible values [Rieffel and Pollak 2000]. Thus, in quantum systems "the amount of parallelism increases exponentially with the size of the system" and an exponential increase in the parallelism is achieved by a linear increase in the amount of space needed [Rieffel and Pollak 2000]. Since quantum computers provide such an exponential speed-up in computational power, it is anticipated that some algorithm will eventually obliterate AES and DES or any other standard based on computational difficulty such as DES and AES. [MagiqTech. Backgrounder, 2002]

One–time pad is perfectly secure but it needs a new key each time it is used and that key would be destroyed afterwards so the key distribution might be a problem.
In general, the presence of a key that is shared only between both the sender and the receiver is crucial for symmetric key cryptosystems to be secure; moreover, the choice of the cipher to be used is very crucial because many are vulnerable to cryptanalysis attacks

like frequency attacks as in the case of substitution ciphers and differential cryptanalysis attacks that could be used against DES.

2.3 Public Key Cryptosystems

Public key cryptography was first proposed in 1976 by Whitfield Diffie and Martin Hellman from the Stanford University [Diffie and Hellman 1976] and its first actual implementation was in 1978 when Ronald Rivest, Adi Shamir and Leonrad Adelman from the Massachusetts Institute of Technology (MIT) developed the RSA cryptosystem.

In Public key cryptosystems (also called asymmetric key cryptosystems) the key used for encryption is different from the key used for decryption and the decryption key cannot be calculated from the encryption key in a reasonable amount of time [Schneier 1996]. If Bob wants to be able to receive message encrypted with public key cryptography, then he must create a private key which he keeps secret and compute a public key from this private key. He then distributes the public key to whomever wants to send him a message so if Alice wants to send a message, she encrypts it with his public key and transmits it to Bob. Bob decrypts the message using his private key and reads it whereas any eavesdropper Eve won't be able to decrypt it without having Bob's private key.

The public key cryptosystems depend on one-way-functions that are defined as follows: "A function F from a set M to a set E is called one-way function if we can easily compute F, but it is computationally infeasible to compute F^{-1}" [Molin 2001].

Definition of public key cryptosystems: A cryptosystem consisting of a set of enciphering transformations E_e and a set of deciphering transformations D_d is called a public key cryptosystem if, for each pair (e,d), the enciphering key e (the public key), is publicly available while the deciphering key d (the private key is kept secret). It must be computationally infeasible to compute private key d from public key e [Molin 2001].

Public key cryptosystems don't substitute symmetric key cryptosystems but they only encrypt keys that are later used by the symmetric key cryptosystems to encrypt the message itself. The reasons for not encrypting messages using public-key cryptography are that it's slow as they are 1000 times slower than symmetric key cryptosystems [Schneier 1996].

In public key cryptography, it is possible for a cryptanalyst to intercept the messages sent between the two communicating entities in the initial key transfer stage and thus he/she can impersonate them and convince each of them that the cryptanalyst is the other entity. In order to overcome this, and be able to verify that a message sent from a person is really a message written by him and not by another person impersonating him, digital signatures are used.

A digital signature is a digital data string that associates a message to its sender [Molin 2001].

The most famous of the public key cryptosystems is the RSA cryptosystem.

2.3.1 The RSA Public Key Cryptosystem

The RSA algorithm generates two large prime numbers p and q where $p \neq q$ and p,q are roughly the same size then computes n=pq and $\varphi(n) = (p-1)(q-1)$. After that the algorithm selects a random positive integer e such that $1 <= e <= \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$ (gcd= greatest common divisor) and then computes d where $1 < d < \varphi(n)$ and $ed \equiv 1 \pmod{\varphi(n)}$ .This can be computed using the extended Euclidian algorithm. Then the public key is (n,e) and the private key is d. [Molin 2001]

Encryption of a message m is done by the following equation:

$$c \equiv m^e \qquad \text{(Equation 2.1)}$$

Where m is the message plain text and c is the cipher text

Decryption of any encrypted message is done by the following equation:

$$m \equiv c^d \qquad \text{(Equation 2.2)}$$

RSA depends on the fact that the decryption key d cannot be computed from the encryption key (e) [Molin 2001].

There are other algorithms that are used for public key encryption such as the Rabin Public-key cryptosystem and the ElGamal Cryptosystem and each of them has its own way for the creation of public and private keys and each has its own signature scheme.

There are also public key algorithms that rely in their theory on the subset sum problem and the knapsack problem like the Merkle-Hellman Knapsack Public-key Cipher (Shamir produced an algorithm that breaks it in 1982 in polynomial time) and the Chor-Rivest knapsack problem (the only known secure knapsack cipher at the present time).

2.3.2 Problems with Public Key Cryptosystems

A problem that faces public key cryptography is that there is no mathematical proof that one-way functions exist and there is no real evidence that they can be constructed. However, there are functions that look one-way in that they are easy to compute but there is no known way to compute their reverse up till the moment [Schneier 1996]. Thus, all public key cryptosystems rely for their security on unproven assumptions that could be suppressed by practical or theoretical advances [Gisin et al. 2002].

The RSA cryptosystem relies on the difficulty of factoring large integers (product of two prime numbers). The fact that factoring is a one way function is because the time required to factor a number increases exponentially with the number of digits in that number. Thus, factoring a number that exceeds 1024 digits is infeasible to compute even on high speed computers in a distributed environment ["Quantum Cryptanalysis" 2002]. Not only RSA but also many other algorithms depend on the fact that factoring large integers is a one-way function.
However, a new kind of cryptanalysis (quantum cryptanalysis) is threatening to be able to factor large integers which could cause the collapse of RSA and many other public key encryption schemes.

Quantum cryptanalysis is based on the existence of quantum computers as it takes advantage of their nature to succeed in what classical computers failed to do ["Quantum Cryptanalysis" 2002]. The power of quantum computation derives from "the exponential state spaces of multiple quantum bits" where a single quantum bit can be in a superposition of 0 and 1 and thus a register of n qbits (quantum bits) can be in a superposition of 2^n possible values [Rieffel and Pollak 2000]. Thus, in quantum systems "the amount of parallelism increases exponentially with the size of the system" and an exponential increase in the parallelism is achieved by a linear increase in the amount of space needed [Rieffel and Pollak 2000].

In 1994, Peter Shor described a quantum algorithm for factoring integers in polynomial time; moreover, he proposed another algorithm for calculating discrete logarithms too [Shor 1994, Shor 1997]. Thus, when (or may be it should be said "if") quantum computers are built, most of the public key cryptography would be unsafe. It's worth noting here that quantum computers with only 3 bits have been built successfully [Rieffel and Pollak 2000].

Moreover, Daniel Bernstein at the University of Illinois at Chicago devised a way to build circuits for factorization with present technology. This means that in the near future RSA and many other public cryptography algorithms will be vulnerable to attacks using these circuits. Many scientists speculate that the NSA might already have built such specialized chips and that keys even as long as 1024 bits might already be compromised [MagiqTech. Backgrounder, 2002].

Chapter 3

Quantum Computing

&

Quantum Cryptography

Chapter 3: Quantum Computing & Quantum Cryptography

3.1 Quantum Computing

In 1982, Richard Feynman observed that certain quantum effects can't be simulated on our classical computers which made him speculate that computation might be more efficient if it was made using these effects. However, this field of creating quantum computers was developing very slowly because of the problems faced regarding the use of quantum effects in doing computations and the uncertainty of whether using quantum effects would actually lead to a speed-up in computation [Rieffel and Pollak 2000]..
When Peter Shor described his algorithm for factoring large integers in polynomial time using quantum computing in 1994, the field got a big boost forward. It became a known fact that computers based on quantum effects would outperform the classical computers present today and customized algorithms, like Peter Shor's one, could be developed to solve many of the problems that were deemed computationally unfeasible before. This started a lot of research activity aimed at building quantum computers and building other quantum algorithms that would solve today's unsolvable problems on quantum computers.

3.1.1 The power of Quantum Computing

In classical computers, we can decrease the time to perform a computation by using parallel processors, but to achieve an exponential decrease in time we need to make an exponential increase in the number of processors which ends up being very expensive. The power of quantum computing is that the amount of parallelism increases exponentially with the linear increase in the number of processors. This effect is called "quantum parallelism" [Rieffel and Pollak 2000]..
This effect derives from the fact that a single quantum bit (qbit) "can be in a superposition of 0 &1"; thus, a register of n qbits would be in a "superposition of all 2^n possible values". These states that are not present in classical registers and computers lead to the exponential size in the quantum state space [Rieffel and Pollak 2000]..

However, there is a problem facing quantum computation. In spite of the fact that quantum systems can perform huge parallel computations, access to the results of the computation is limited by the rules of quantum physics. Reading the results of the computation is done through making a measurement to get the required value. However, Heisenberg uncertainty principal states that "measuring a quantum system disturbs it and yields incomplete information about its state" [Brassard and Crepeau 1996], which means that making the measurement to read the required value would disrupt the quantum state and make it impossible to read other values later on. This is a severe problem as it makes us able to read only one value of the results of a parallel computation disregarding all the other results. This problem is the major problem facing quantum computing that has been hindering the research and production on quantum computers.

Some techniques have been devised to get over this problem of measuring the results of a quantum computation in order to exploit the power of quantum parallelism. One such technique manipulates the quantum state so that a common property of all the output values such as symmetry or period can be read off. This technique depends more on the algorithm that is run on the quantum computer as it must be customized so that it would have the results depending on this period. This is the technique that Peter Shor has used in his algorithm for factoring large integers. Another technique for overcoming the problem is to transform the quantum state and it was used in a search algorithm proposed by Grover [Rieffel and Pollak 2000].

As noted before, Peter Shor has developed an algorithm that is capable of factorizing large prime numbers using quantum computation on quantum computers. Moreover, Lov Grover developed a technique for searching a list of unordered n items in $O(\sqrt{n})$ on a quantum computer which is more efficient than doing the same thing on a classical computer (On classical computers searching such an unordered list is $O(n)$) [Grover 1998]. It's not fully known the variety of applications that would be able to benefit from the power of quantum computation; however, it might be able to solve NP-complete problems in polynomial time if clever algorithms are developed for that like Peter Shor's algorithm on factoring large prime numbers and Lov Grover's algorithm on searching an unordered list. Whether quantum computers would be able to solve NP-complete problems or no is an open-question at the moment [Rieffel and Pollak 2000].

3.1.2 Building Quantum Computers

There have been many proposals for building quantum computers based on different techniques; however, all those techniques face scaling problems. Whether or not useful quantum computers would be built is still an open question that nobody is sure of. A breakthrough is needed to build quantum computers with hundreds qubits rather than tens of qubits [Rieffel and Pollak 2000].

There are four techniques proposed for building quantum computers which are optical, solid-state, ion trap and nuclear magnetic resonance (NMR) techniques. The optical and solid-state techniques show promise, but NMR and ion trap are the most advanced ones.

In an ion trap quantum computer, "a linear sequence of ions representing the qubits is confined by electric fields. Lasers are directed at individual ions to perform single bit quantum gates. Two-bit operations are realized by using a laser on one qubit to create an impulse that ripples through a chain of ions to the second qubit, where another laser pulse stops the rippling and performs the 2-bit operation. The approach requires the ions be kept in extreme vacuum and extremely low temperatures". [Rieffel and Pollak 2000]

The NMR approach is the most advanced one and it is better than the ion trap technique as it works in normal room temperature (rather than needing extremely low temperatures and vacuum as the ion trap technique). The NMR approach uses "macroscopic amount of matter and encode a quantum bit in the average spin state of a large number of nuclei. The spin states can be manipulated by magnetic fields and the average spin state can be measured with NMR techniques" [Rieffel and Pollak 2000]. The problem with this approach is that it doesn't scale, but a proposal has been made to overcome this problem [Schulman and Vazirani 1998].

Quantum computers with 3 qbits have already been built using the NMR technique.

The major problem that faced quantum computers in the past was the distortion of the quantum states due to interaction with the environment. Many people thought that quantum computers can't be built because it would be impossible to isolate them enough from the external environment. The solution to this problem came from a rather an unexpected angle as the solution came from the algorithm side rather than the physical side. Quantum error correction techniques were the solution to this problem and it became possible to design quantum error correcting codes that detects errors and enables the reconstruction of the exact error-free quantum state. [Rieffel and Pollak 2000]

3.2 Quantum Cryptography

Quantum Cryptography is the newest branch of cryptosystems and it's the hottest topic in cryptography now especially that quantum cryptanalysis threatens the public key cryptosystems. It is based on a combination of the concepts of quantum physics and information theory. Quantum cryptography isn't used to encrypt the message itself but it is rather used for the generation and distribution of random secret keys between the sender and the receiver. The key created is used with symmetric cryptosystems (typically one-time pads) to encrypt the messages between the sender and the receiver. Thus, quantum cryptography is a technology employing a combination of quantum mechanical phenomena and classic cryptographic schemes with the goal of generating a secret key (or extending a short key) between two communicating parties [Kollmitzer et al. 2002].

It has been proven through several researches of the topic that key distribution via quantum cryptography is secure against "an eavesdropper with unlimited computing power" which is a thing that public and symmetric key cryptosystems do not have and are not expected to have [Boyer et al. 2001].

3.2.1 Scientific Basis

Quantum Cryptography is based on the Heisenberg uncertainty principle which states that "measuring a quantum system disturbs it and yields incomplete information about its state." Thus, eavesdropping on the quantum communication channel causes unavoidable disturbance that can be recognized by the legitimate users [Brassard 1994]. This helps in creating a system for key distribution between two people who don't initially share any secret information. This cryptosystem is secure against eavesdropping even if the Eavesdropper has unlimited computing power. Once the secret key is established, classical cryptographic techniques (like the one-time pads) can be used to allow the secure exchange of information.

The steps for the quantum key distribution protocol are as follows:

1. Creation of a key between the two communicating parties (Alice and Bob) using one of several techniques such as polarization methods or phase methods (explained in the next sections).
2. Error Correction for the errors and discrepancies between the keys with Alice and Bob.
3. Privacy Amplification which is concerned with reduction of the eavesdropper information that might be gained during the initial creation of the key.

These steps are explained in further details in chapter 4.

In classical cryptography (symmetric and asymmetric), the communication between Alice and Bob can always be passively monitored by an eavesdropper which means that Eve can be capturing all the data sent between the communicating parties without being noticed by Alice and Bob. However, in quantum cryptography this is not possible because in quantum key distribution the data is sent as a polarization of photons or as a shift in the phases of the photons and any eavesdropper cannot even gain partial information on the state of the photons holding the data without disturbing the system in a random and uncontrollable way which would be evident to both Alice and Bob [Kollmitzer et al.2002]

At present, the channel used for communication is fiber optics; however, this has the limitation of being limited to no more than 120 km as long as quantum repeaters are not available and that's why there are experiments on using free space in the quantum channel as will explained later.

There are two possibilities for wavelengths for use by quantum cryptography within the fiber optic communication channel. The first wavelength is around 800 nm and the second is the range of 1350-1550 nm. There are efficient photon counters (needed for the detection of the photons) commercially available for the wavelength of around 800 nm; however, this is not compatible with present telecommunications optic fibers and so would need special fibers. On the other hand, the wavelength of around 1350-1550 nm is compatible with the present optic fibers but would need the development of new photon detectors [Kollmitzer et al.2002].

The source of quantum signals can be either single photon source (faint laser pulse) or entangled photon pairs (There is a photon and a photon pair).

The coding of the information can be polarization coding or phase coding that will be described in later sections with more detail.

3.2.2 History of Quantum Cryptography

Quantum Cryptography was born in the late 1960s by Stephen Weisner who wrote "Conjugate Coding" which was unpublished and unnoticed by the time. Stephen Weisner had the concept of "Quantum money" which is impossible to counterfeit. His idea was to "charge" the dollar bill with several photons that are polarized in two non-orthogonal bases which means that measuring the state of the photons using one basis randomizes the value of the other. Thus, a counterfeiter will have to measure the state of the photons of a dollar to be able to reproduce them in the new counterfeited bill. However, since he doesn't know the original basis that was used to code the photons (this is kept a secret by the bank), he'll have to try either of the bases which will necessarily randomize the other. Thus, the counterfeiter will have a 50% error. On the other hand, the bank has the basis that was used to code the photons and so can measure the polarization of the photons on the dollar bill and can check whether the bill is counterfeited or not. This idea was real novel but it was impractical because it is impossible to store a photon trapped for a long period of time [Brylevski 2002].

In 1979 Charles Bennet and Gilles Brassard thought of using the photons to transmit information through a quantum channel rather than storing it as Weisner proposed. The quantum channel consisted of optical fiber and the transmission was done by light pulses so weak that the probability of a photon appearing in light pulses is lower than 1 per pulse. Their aim was to provide two communicating parties Alice and Bob with an identical sequence of random bits that can be used to encrypt messages between them.

However, in 1982 Quantum Cryptography had a major drawback during the CRYPTO '82 conference which made the people have the impression that everything that has to do with quantum cryptography was unrealistic [Bennet et al. 1992].

In the past few years, there was a remarkable interest in quantum cryptography and it became part of the mainstream computer science and physics especially after the paper "Experimental Quantum Cryptography" published in 1991 by Charles Bennet (IBM Research), Gilles Brassard (University of Montreal) and François Bessette (University of Montreal) [Bennet et al. 1992] and the experiment mentioned in that paper. That experiment was performed at IBM and was the first demonstration of Quantum Cryptography over 30 cm of air with polarized photons [Zbinden et al. 2000] and marked the start of experimental improvements during the subsequent years.

In 1996, Antoine Muller at the University of Geneva used a single photon polarization based systems at wavelength 1300 nm to create a key at a distance of 23 km between the cities of Geneva and Nyon. The interesting feature is that they used the standard optic fiber used by Swisscom for carrying phone conversations as their quantum channel [Kollmitzer et al. 2002, Muller 1996]. This was the first quantum key distribution experiment to be done outside physics labs and had a strong impact on the interest of the public in the field of quantum cryptography. That experiment was done after he made another experiment to implement the quantum key distribution protocol at a distance of 1100 meters using a wavelength of 800nm [Gisin et al. 2002].

Paul Townsend of BT Laboratories was able to use standard telecommunication fibers to create a key at a distance of 10 km using single-photon polarization encoding at 800 nm. He was able to do that by carefully controlling the launching conditions of the photons [Kollmitzer et al.2002].

At the University of Innsbruck, Anton Zeilinger and his group were the first to use entangled-polarization based system over a distance of 720 meters using special optic fibers that were designed to operate at wavelength 700 nm [Kollmitzer et al.2002].

3.2.3 Quantum Key Distribution coding schemes

All existing encodings for bits in the quantum key distribution systems use non-orthogonal states of photons as carriers of information, as they cannot be cloned (duplicated) by an eavesdropper. Even if imperfect cloning is attempted, it induces errors in the quantum transmission. That's why any two non-orthogonal states can be used for quantum cryptography [Bennet 1992].

3.2.3.1 Quantum Key distribution based on Photon Polarization

The essential quantum property involved in this key distribution scheme is the existence of pairs of properties of photons that are incompatible "in the sense that measuring one property randomizes the value of the other". Any pair of polarization states is referred to as bases if they correspond to a measurable property of a single photon and two bases are said to be conjugates if measuring one of them randomizes the other.

The quantum key distribution protocol described by Bennet and Bassard used two conjugate bases which are the rectilinear basis (horizontal vs. vertical polarization) and circular basis (left vs. right circular polarization). They are referred to as canonical bases. Another basis that is sometimes used with these bases consists of 45 and 135 degrees polarization which is conjugate to the other bases too.
This scheme became the first experiment on Quantum key distribution over 30 cm using free air optical path as quantum channel [Bennet et al. 1992].

3.2.3.2 Problems of Photon Polarization

Long optical fibers induce a polarization transformation for the transmitted photons. This polarization transformation remains stable and low for a while (which is bearable and doesn't cause problems), but it then suddenly increases indicating a modification of the polarization transformation in the fiber. This means that active alignment would be needed to compensate for this which is actually possible but very difficult. Such an alignment system has already been implemented by James Franson [Franson and Jacobs 1995], but he didn't complete his work and pursue on that direction [Brylevski 2002].

3.2.3.3 Coding based on Phase Coding

Because of the polarization transformation problem in photon polarization coding in optical fibers, it is very difficult to setup quantum key distribution along long distances using them. The other option is to encode the value of the bits in the phase of the photons rather than their polarization [Brylevski 2002].

In Phase coding, Alice can apply a phase shift (0, Π/2, Π, 3Π/2) to encode a bit value. Phases 0 and Π/2 encode bit value 0, while phase value Π , 3Π/2 are used to encode bit value 1. On the other hand, Bob applies a basis choice by randomly applying and phase shift of 0 or Π/2.

A table for the four states phase coding is shown below [Brylevski 2002]

Table 3-1 Phase Coding States

Alice		Bob		
Bit Value	φ_A	φ_B	$\varphi_A - \varphi_B$	Bit Value
0	0	0	0	0
0	0	Π/2	3Π/2	?
1	Π	0	Π	1
1	Π	Π/2	Π/2	?
0	Π/2	0	Π/2	?
0	Π/2	Π/2	0	0
1	3Π/2	0	3Π/2	?
1	3Π/2	Π/2	Π	1

Thus, photon phase coding can be used in the quantum cryptography protocol just the same as photon polarization.

3.3 Problems of Quantum Cryptography

A serious limitation of Quantum cryptography using fiber optics as communication channel is that it can't extend more than 120 kilometers because there are no quantum repeaters available; however, this should be solved by using free-space as a quantum channel (using free-space as the quantum channel is detailed in the next section).

In quantum cryptography eavesdropping is indistinguishable from the noise to Alice and Bob as there are no practical means to distinguish an eavesdropper attack from noise available in present photon counters and quantum communication channels [Castelletto et al. 1995]. Noise and eavesdropping can cause the secure quantum exchange to fail. This leads to 2 potential problems [Ford 1996]:

1. An eavesdropper can prevent a communication.
2. If attempting to make the key distribution in the presence of high noise, then the eavesdropping attempts would be more feasible.

Moreover, the noise indicated above is putting limits on the distances between the two communicating parties, Alice and Bob [Zbinden et al. 2000].

Another problem that needs to be solved for quantum key distribution is the wavelengths to be used for the fiber optics (if the fiber optics are used as communication channels). There are two possibilities for wavelengths for use by quantum cryptography within the fiber optic communication channel which are around 800 nm and the range of 1350-1550 nm. There are efficient photon counters (needed for the detection of the photons) commercially for the wavelength of around 800 nm; however, this is not compatible with present telecommunications optic fibers and so would need special fibers. On the other hand, the wavelength of around 1350-1550 nm is compatible with the present optic fibers but would need the development of new photon detectors.

Another problem that was mentioned is the polarization transformation for the transmitted photons in polarization coding which means that the quantum key distribution systems have to actively compensate for the polarization changes.

3.4 Free-Space as the Quantum Channel

In the previous stated experiments and work done, fiber optics were used as the communication channel; however, a serious limitation of using fiber optics in quantum cryptography is that it can' extend more than 120 kilometers because there are no quantum repeaters available. Thus, there are now initial experiments for using free space systems as quantum channels (i.e, photons are sent between telescopes). The goal is to use quantum cryptography between satellites. This free space communication has been demonstrated for distances of several kilometers and theoretical estimates indicate that single photon communication between earth and satellites is feasible. [Kollmitzer et al. 2002]

Transmission over free space has some advantages over transmission in fiber optics. The first is the distance limitation stated before because fiber optics as a channel provide a limit on the distance between the two communicating parties Alice and Bob (unless quantum repeaters are invented); however, there is no theoretical limit on the distance between the communicating parties in case of free space communication. Another advantage for using free space as the quantum channel is that the atmosphere has a "high transmission window at a wavelength of around 770nm" which means that existing commercial photon detectors could be used for photon detection. This means also that a high channel transmission speed over long distances can be ensured. Moreover, "atmosphere is only weakly dispersive and essentially non-birefringent at these wavelengths" which means that it won't alter the polarization of a photon. [Gisin et al. 2002]

On the other hand, there are drawbacks for using free air as the communication channel. In contrast to fiber optics, which is a guided medium, energy transmitted in free-space spreads out leading to higher transmission losses; moreover, daylight and moonlight at night can "couple in the receiver" increasing the error rate. Finally, it's clear that free-space transmission performance would depend on the atmospheric conditions and would only be possible in clear weather [Gisin et al. 2002].

There were many experiments already done using free-space as the quantum channel. In 1996 Jacobs and Franson were able to conduct the first free-space quantum cryptography experiment over a distance of a few centimeters; after that they exchanged a key over a distance of 150 m in standard florescent lightening, then over a distance of 75 m in outdoor bright daylight [Gisin et al. 2002]. Later, Buttler and Hughes and their team were able to conduct experiments on distances of 1 km in outdoor nighttime conditions [Buttler et al. 1998] and 1.6 km in daylight conditions [Buttler et al. 2000]. Finally, Gormen and Rarity were able to make the key distribution over a distance of 1.9 km in nighttime [Gorman 2001].

A quantum cryptography link could be established to the low orbit satellites in order to be able to exchange keys between any two parties (wherever they are) through these satellites. In that case, each of Alice and Bob (the communicating parties) would exchange a key with the same satellite independently to produce two keys K_A and K_B. After that the satellite computes $K = K_A$ XOR K_B and announce the result publicly; thus, Bob can know K_A since he knows both K and K_B whereas nobody else can know the value of K_A which could then be used as a secure key used between Alice and Bob. The main problem in this scheme is due to the fact that satellites move with respect to the ground which generates a problem in beam pointing to the satellite. The major disadvantage of this protocol for key exchange between Alice and Bob through the satellite is the fact that the satellite operator knows the key between Alice and Bob.

3.5 Current Projects in Quantum Cryptography

3.5.1 The Arc Project

Kollmitzer, Monyk and Suda from the Austrian Research Center (ARC), Seibersdorf Austria are now building upon the knowledge of Prof. Zeilinger's group (their experiments are noted in section 3.2.2) with the aim of developing a "system relevant for practical application which uses quantum cryptography to generate and exchange keys for subsequent communication over public channels" [Kollmitzer et al. 2002]. The scheduled period of their project is 5 years at the end of which they should have developed an industrial quantum cryptographic prototype that will generate and exchange absolutely secure keys between two communicating parties. They aim for their developed prototype to be ready for transfer to industrial serial-production

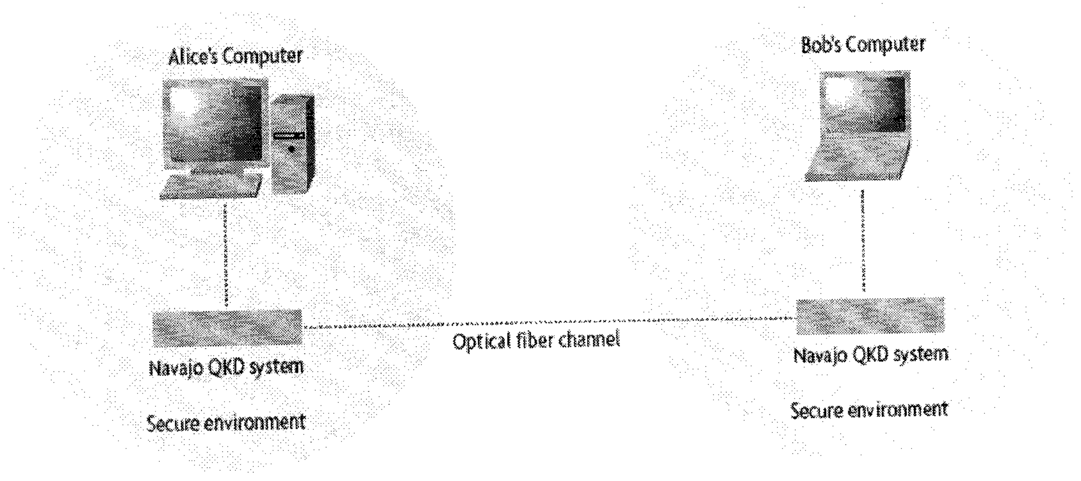
The project team is planning to use optical fibers as their communication medium, polarization coding with entangled photons as their coding scheme and they'll use a wavelength of 800nm. The use of the 800 nm wavelength means that they'll need special optical fibers and won't be able to use ones commercially available right now. They could shift to the use of telecommunication optical fibers commercially available if that is technically feasible; however, at the moment it's not.

Quantum Cryptography is limited to distances less than 120 km (actually, the longest distance that an experiment reached is 23 kilometers) and thus the project team will have to focus on concepts and ways to make the exchange of the key feasible over longer distances.

3.5.2 Magiq Navajo Project

By the end of 2003, MagiQ Technologies (www.magiqtech.com) has announced its product, Navajo, which is a quantum cryptographic solution offering encryption based on quantum cryptography. An overview of the Najavo system is shown in figure 6.1 [MagiqTech Navajo White Paper 2003]

Figure 6-1 Overview of Najavo System



Najavo is the first commercial product that applies quantum cryptography to be able to secure transmissions. It addresses the key distribution and secure key exchange problems; moreover, it gives the chance of refreshing the keys at a high rate which is desirable to keep the keys secure.

Najavo tries to achieve the following objectives:

1. It is capable of detecting intrusions. The system would be able to detect any eavesdropping attempt.
2. It is capable of protecting against theft of key information. This is done through the rapid key refresh rate of the keys between the sender and the receiver; thus, the keys stolen are of no use.
3. It is proven secure and it can withstand the advance of technology as the security of the system is established by the rules of quantum physics. [MagiqTech Navajo White Paper 2003]

3.5.3 Other Projects

Another project is running now at the NTNU (Norges Teknisk Naturvitenskapelige Univestet) university in Norway where many masters projects are about quantum key distribution and they aim at generating a key transmission software. [Brylevski 2002]

There are also projects going on with quantum cryptography at the Los Alamos laboratory, Munich, Geneva, Oslo and the University of Vienna.

Chapter 4

The BB84 Protocol

Chapter 4: The BB84 Protocol

BB84 is the protocol widely used in Quantum Key distribution. It is the protocol detailing the steps to be applied to achieve the goal of quantum key distribution which is creating a secure key that is shared between the 2 communicating parties (Alice and Bob). The BB84 protocol was proposed in 1984 by Bennet and Brassard [Bennet and Brassard 1984]. Following is a brief description of the BB84 protocol along with a description of the Bennet and Brassard Experiment.

The steps for the quantum key distribution protocol whatever the encoding are as follows:

1. Creation of a key between the two communicating parties (Alice and Bob) using one of several techniques such as polarization methods or phase methods.
2. Error Correction for the errors and discrepancies between the keys with Alice and Bob that were created in step 1.
3. Privacy Amplification which is concerned with reduction of the eavesdropper information.

Before describing the protocol algorithm and the steps that need to be applied for the implementation of the BB84 protocol, it is important to give an overview of random number generation as it is crucial for the first phase of the BB84 protocol (the creation of the key phase).

4.1 Random number generation

Computers are deterministic systems and thus they don't actually create truly random numbers, but they produce pseudo-random numbers in general. A quantum solution for this problem is to rely on the "random choice of a single photon at a beamsplitter" [Rarity et al. 1994]. The randomness of the numbers using this technique would be guaranteed by the laws of quantum mechanics; however, care should be taken so as not to introduce "artifacts that could correlate adjacent bits" [Gisin et al. 2002]. There have been some experiments on this technique and there are prototypes available for it.

4.2 Protocol Algorithm (based on photon Polarization)

4.2.1. Creation of the key

1. Alice sends Bob a random sequence of the four canonical kinds of polarized photons to Bob.
2. Bob chooses randomly for each photon whether to measure the photon's rectilinear or circular polarization and announces publicly which kind of measurement he made for each photon (but doesn't announce the results of his measurement).
3. Alice tells Bob publicly whether he made the right measurement (for each photon).
4. Alice and Bob discard all bit positions where Bob has performed the wrong measurement. They also discard the bit positions where Bob didn't detect anything at all.
5. For the remaining photons, horizontal or left-circular polarizations are interpreted as 0 whereas vertical and right-circular polarizations are interpreted as 1. The resulting is a binary string that is shared between Alice and Bob, provided that Eve didn't eavesdrop on the quantum communication channel.

After that Alice and Bob test for eavesdropping by comparing the polarizations of random subsets of photons (Note that measuring the value of the polarization by an eavesdropper would make a difference in the state of the photon and thus Bob wouldn't read the value the same as Alice) [Bennet et al. 1992].

4.2.2. Error Correction

Presence of noise in the quantum channel and photon detectors induces errors even in the absence of an eavesdropper. Thus, errors are induced either due to the presence of an eavesdropper or due to noise in the detectors.
This problem of errors can be solved using the following protocol that reconciles the differences between sent and received data in the quantum transitions and distills from it a smaller amount of data that is perfectly secret (It is important to note that the next steps

are done using the normal communication channels and not quantum channels as the "creation of keys" stage):

1. Alice and Bob agree on a random permutation of the bit positions in their strings and partition the strings into blocks of size k that is unlikely to contain more than one error
2. Alice and Bob compare the parities of each block, if they are the same the data in the block is matching, if not then the block is further divided to block and parities compared till the error is detected and corrected
3. Alice and Bob discard the last bit of each block after reconciliation to avoid the leaking of information assuming that Eve has been eavesdropping during the reconciliation process

Steps 1-3 are repeated several times with different permutations and increasing block sizes until Alice and Bob are sure that the shared data has no errors [Bennet et al. 1992].

4.2.3. Privacy Amplification

Privacy amplification is concerned with distilling "highly secret" shared information from a larger amount of shared information that is partially a secret. Using BB84 protocol, Eve might have been able to spy on some of the information sent on the channel and thus might be able to know the value of some part of the shared key between Alice and Bob [Bennet et al. 1995].

Assuming that x is the shared string between Alice and Bob and that it consists of n bits. Assuming also that Eve's knowledge is no more than L bits where $L<n$. It has been shown by Brassard and Bennet that a hash function h can be publicly chosen form a set of appropriate class of functions that will map n bits to n-L-s bits such that Eve's expected information would be less than $\frac{2^{-s}}{\ln 2}$ [Bennet et al. 1992].

An example for doing privacy amplification would be for Alice to randomly choose a pair of bits, compute their XOR and announce which bits she chose (not the result value of her XOR operation). Thus, Bob can compute the XOR value too and both Alice and Bob would replace the two bits which were XORed by their XOR result. In this way, the shared key will shorten but will still be kept error free; however, if Eve knew partial information on the two bits, then her information would be even less after privacy amplification. For example, if Eve knew the value of one bit but she didn't know the value of the other, then she would have no information at all about the XOR result value. If Eve knew the value of both bits with probability 60%, then the probability of her guessing correctly the XOR result value would be 52%. This XORing of values would be repeated several times in order to minimize (or even diminish) what Eve knows of the shared key.

A problem in this protocol is that Eve can impersonate both Alice and Bob for the other (called the man-in-the-middle-attack) and thus she would end up with a string shared with Alice and a string shared with Bob [Bennet et al. 1992 , Ford 1996]. This can be solved by using an authentication scheme to certify that Alice is talking to the real Bob and vice versa which means that Alice and Bob need to have some shared secret information beforehand to authenticate with and that each time a few bits of this key is unfit for re-use again (this is actually easy because each instance of the key distribution protocol provides Alice and Bob with a large amount of new shared key information. Thus, this protocol in this case would be key expansion protocol rather than key distribution protocol [Bennet et al. 1992].

This BB84 protocol has been assumed to be unconditionally secure by Dominic Mayers which means that a "security result is expected to hold against all attacks allowed by quantum mechanics." [Mayers 2001]

4.3 Eavesdropping Strategies

Eavesdropping is concerned with finding protocols which would allow Eve to know the secret shared key and still Alice and Bob think that their shared key is totally secure. There are two main eavesdropping strategies for this quantum key distribution encoding scheme which are intercept/resend and beam splitting

4.3.1 Intercept/Resend Attack

Let μ be the expected number of photons per light pulse and it should be sufficiently smaller than 1.

In this attack an eavesdropper Eve intercepts selected light pulses and reads them in a basis that she chooses. For each pulse that she intercepts, with probability μ, Eve would be able to successfully detect a photon (assuming she has perfect photon detectors). Then, Eve would fabricate a new pulse with the same polarization as she detected and sends it to Bob.

In classical communication channels, Eve would be able to measure Alice's signal exactly with no probability of error and then send an exact copy to Bob without being detected at all. However, in this quantum key distribution scheme Eve doesn't actually know the basis in which Alice has polarized her photos so can't read information deterministically and then send an exact copy of it to Bob. It has been proven that Eve will introduce at least 25% error in the pulses detected by Bob if she uses this kind of attack [Bennet et al. 1992].

Therefore, if Alice and Bob discover t errors in their raw transmission, then they should assume that Eve have intercepted $4t + 5\sqrt{12t}$ of their bits (The second term is an allowance for statistical errors). Actually, these errors can result in practice from noise at the detectors, disturbance in the quantum channel or optical misalignment but it's safer for Alice and Bob to assume that all errors are due to intercept/resend attacks.

4.3.2 Beamsplitting Attack

This attack depends on the fact that transmitted light pulses are not pure single photons. Thus, Eve diverts a fraction f of the original beam to herself letting the remainder pass undisturbed to Bob. This attack doesn't introduce errors as the intercept /resend attack but it reduces the intensity reaching Bob by 1-f [Bennet et al. 1992] which might not be detectable by Alice and Bob.

This attack could leak each bit to Eve with probability μ (where μ is the expected number of photons per light pulse) and assuming that there are N pulses in the transmission then Eve would learn less than

$$N\mu + 5\sqrt{N\mu\,(1-\mu)}$$

Bits through this kind of attack (where the second term is an allowance for statistical errors) [Bennet et al. 1992].

Therefore, Alice and Bob can determine probabilistically the amount of data leaked to Eve. Following is a summary of the experimental results of the experiment done by Bennet and Brassard.

4.4 Experimental Results

These are the experimental results obtained by Bennet and Brassard from the experiment they did using photon polarization as an encoding scheme.

4.4.1 A run without Eavesdropping

The intensity used, μ, was = 0.12 and 715,000 pulses were sent from Alice to Bob of which 2000 were successfully received by Bob in the correct basis. The string with Bob actually contained 79 errors (this is due to noise) which mean that the error frequency was 3.95%. Next, the block parity comparison has been performed on random permutations of the strings at Alice and Bob's sides to reach to 1379 identical bits and discovering 76 errors (By interpolation Alice and Bob estimate 3.6 errors eliminated without detection). Thus, it was estimated that 466 bits could have been leaked to Eve (226 from intercept/resend attacks and 240 from beamsplitting attack). The string is then compressed by random subset hashing to make it just 754 bits whose Eve's expected information is less than 10^{-6} bit.

4.4.2 A run with Eavesdropping

In this experiment intercept/resend attacks has been attempted on 1/8 of the pulses whereas beamsplitting has been attempted on all the pulses.

The intensity used, μ, was = 0.12 and 715,000 pulses were sent from Alice to Bob of whom 2000 were successfully received by Bob in the correct basis. Due to the intercept/resend attacks Bob had 160 errors with an error frequency 8% (Actually during this experiment Eve learned 336 bits in total).

Next, the block parity comparison has been performed on random permutations of the strings at Alice and Bob's sides to reach 1007 identical bits and discovering 148 errors (By interpolation Alice and Bob estimate 14 errors eliminated without detection). Thus, it was estimated that 699 bits could have been leaked to Eve (459 from intercept/resend attacks and 240 from beamsplitting attack). The string is then compressed by random subset hashing to make it just 105 bits. This compression that removed 902 bits "was based on a very conservative estimates of what Eve might know [Bennet et al. 1992].

4.5 Variations of the BB84 Protocol

There are many variations proposed to the described BB84 protocol. One such variation is for the bases not to be chosen with equal probability (i.e. there is some bias towards one of the bases). This would mean that the probability of Alice and Bob choosing the same basis would higher than 0.5 which is the probability if both bases would be chosen with the same probability. On the other hand, this would put Eve in a better position as she could then make more guesses that are likely to be true if she knew which basis is with the higher probability.

Chapter 5

An Enhanced Algorithm for the BB84 Error Correction Phase

Chapter 5: An Enhanced Algorithm for the BB84 Error Correction Phase

5.1 Standard Error Correction Phase

As described before in chapter 4, the BB84 protocol consists of 3 phases which are

1) Creation of a key between the two communicating parties
2) Error correction for the discrepancies in the keys between Alice and Bob
3) Privacy amplification to reduce possible eavesdropper information.

The phase that this research concentrates on is the error correction phase, as this research aims at enhancing the error correction phase in order to reduce the number of bits lost during this phase.

At the start of the error correction phase, each of the two communicating parties (Alice and Bob) has a key that resulted from the first phase of the BB84 protocol. Ideally, the key that Alice and Bob have should be the same; however, practically noise distorts many transmissions of the first phase of the BB84 protocol and thus discrepancies between the keys with Alice and Bob occur. [Noise is not the sole reason for the discrepancies as the presence of an eavesdropper induces errors in the keys too]. The error correction phase is required to remove those discrepancies. At the end of this phase, Alice and Bob should have an identical key; moreover, they should have an error rate estimate to be able to know the percentage of discrepancy approximately. According to the error rate, the BB84 protocol goes to the next phase which is privacy amplification; however, if the error rate is too high, it means that an eavesdropper has been listening to the communication and the thus the protocol should be aborted and restarted again.

Thus, the error correction phase takes as its input two keys which are the keys that Alice and Bob have at the end of the Creation of the key phase. At the end of the error correction phase, an identical key should be reached between Alice and Bob and they both should have the error rate.

The error correction phase consists of the following steps:

1. Alice and Bob agree on a random permutation of the bit positions in their strings and partition the strings into blocks of size k that is unlikely to contain more than one error.
2. Alice and Bob compare the parities of each block, if they are the same the data in the block is matching. If not then the block is further divided to smaller blocks and parities compared till the error is detected and corrected.
3. Alice and Bob discard the last bit of each block after reconciliation to avoid the leaking of information assuming that Eve has been eavesdropping during the reconciliation process

In the error correction phase, steps 1-3 are repeated several times with different permutations and increasing block sizes until Alice and Bob are sure that the shared data have no errors [Bennet et al. 1992].

According to the algorithm, there are two free parameters that need to be supplied to an implementation of the error correction phase. The parameters are the initial block size that would be used to partition the initial key to blocks and the block size increment that should be applied after finishing the 3 steps and starting all over with another loop. The values of the initial block size and the block size increment actually depend on the key value and the errors present in them; thus, we used them in the present implementations as test values that are applied with different values and we have been monitoring their effect on the final outcome of the error correction algorithm.

There is no reliable implementation of the error correction phase on the Internet to be able to use it as a basis for our implementation. However, a full description of the algorithm is described in detail in the "Experimental Quantum Cryptography" paper published in 1992 by Bennet and Barassard [Bennet et al. 1992]. Thus, the implementation of the algorithm was made according to the algorithm described in that paper. That algorithm is the algorithm that has been used for all the later experiments on the Quantum Cryptography topic.

The algorithm is implemented using C# and it ran over the .Net framework. The application has a "Sender" class and a "Receiver" class. The "Sender" class does the steps needed from Alice and the "Receiver" class does the steps at Bob's side. When running an experiment, 2 instances of the application should be run (one instance for Alice and another instance for Bob). The 2 instances communicate in this phase using the normal networking infrastructure (as required by the BB84 protocol) to be able to reconcile on the final key [The 2 instances can be run on different machines as long as a network connection can be established between them]. After reconciling the key with Alice and Bob, an estimate of the error rate is calculated to be used to determine whether to continue or to abort.

5.2 Standard Algorithm Implementation

The standard algorithm implementation is built up of 2 modules which are the “Sender” (Alice) and the “Receiver” (Bob) where each of them does the part needed from the respective user during the error correction phase.

A quick overview of the steps needed of each module is given in this section

5.2.1 Sender (Alice) module

When running the algorithm implementation as Alice, the error correction phase implementation receives an initial key value which represents the key that Alice has. The key is represented by an array of bits. The following tasks are applied:

1. Alice creates a random permutation of numbers from 1 till the size of the key that she has. This permutation is used later to shuffle the locations of the bit values in the key array.
2. Alice sends the permutation created in step1 to Bob. This is done through TCP over the network so the “Receiver” (Bob) module could be running on a different machine. By receiving the permutation, Bob and Alice would have the same permutation and could use it to shuffle the locations in the key array if they want to. This permutation would be later used to shuffle the order by which the bits are compared.
3. Alice computes the number of blocks that it would split the key to. The number of blocks is calculated to be equal the number of bits in the key divided by the current block size. [The current block size is equal to the initial block size at the start. The initial block size is one of the parameters that are set for the error correction phase].
4. Alice partitions the key into blocks each of the size of the current block size (the current block size is the initially the initial block size that is a parameter to the error correction phase). The bit positions of the each block are taken from the key and are computed using the permutation created in step 1.The number of blocks has been computed in step 3 noted earlier.

5. For each block the following is applied:
    a. The parity of the block is calculated.
    b. The parity computed is sent to the receiver (Bob) through networking.
    c. The parity of the corresponding bit at Bob's side is received.
    d. If the parity is different between the two
        i. The number of errors in the pass is incremented
        ii. The block is split into 2 and the parity of each is checked with the parity on Bob's side. This continues recursively until the exact error location in the block is identified.
        iii. The error location in the block is mapped to the error location in the key and its value is flipped (if it were 0, it would be changed to 1 and vice versa).
6. The last bit of each block is removed from the key so that the parity information sent in step 5 would be useless for any eavesdropper who might have been able to record them. Moreover, a bit is removed for each group whose parity has been sent over the network (this applies to the smaller blocks whose parities are sent on the network in step 5.d.ii)
7. The current block size is incremented with the block size increment.

Steps 1 to 7 are repeated over so that the key would be passed over the error correction phase with different block size comparisons. When the key iterates 20 passes without having any discrepancy between the parities at Alice and Bob's side, it is assumed that the key is now the same with both of them.

At this stage, the error correction phase computes the error rate to check whether it is in the acceptable limits. This is done by applying the following steps:

1. The total number of errors is calculated which is a count of the errors encountered at the different stages of the error correction phase.
2. An estimate of errors that hasn't been encountered is made by interpolation. These errors were undiscovered because they have been removed when deleting the last bit of each block; thus, they were removed but not counted as errors by the algorithm so we need to interpolate them.

3. The total number of errors is calculated as the sum of the counts computed at steps 1 and 2.
4. The error rate is computed to be the total errors divided by the number of bits in the initial key.

If the error rate is too high, then the algorithm aborts because of the hazard of an eavesdropper and the first phase of the BB84 protocol (the creation of the keys) should be re-applied. The error rate induced by the several techniques of eavesdropping would be higher than 11% and thus 11% has been taken the threshold of the error rate. [Gisin et al. 2002].

5.2.2 Receiver (Bob) module

When running the algorithm implementation as Bob, the error correction phase implementation receives an initial key value which represents the key that Bob has. The key is represented by an array of bits. The following tasks are applied:

1. Bob receives the permutation created by Alice. This is done through TCP over the network. By receiving the permutation, Bob will have the same permutation Alice has and could use it to shuffle the locations in the key array if they want to. This permutation would be later used to shuffle the order by which the bits are compared.
2. Bob computes the number of blocks that it would split the key to. The number of blocks is calculated to be equal the number of bits in the key divided by the current block size. [The current block size is equal to the initial block size at the start. The initial block size is one of the parameters that are set for the error correction phase].
3. Bob partitions the key into blocks each of the size of the current block size (the current block size is the initially the initial block size that is a parameter to the error correction phase). The bit positions of the each block are taken from the key and are computed using the permutation received in step 1.The number of blocks has been computed in step 2 noted earlier. The blocks at Bob and Alice side are the same as they applied the same algorithm to create them.

4. For each block the following is applied:
    a. The parity of the block is calculated.
    b. The parity of the corresponding bit at Alice's side is received.
    c. The parity computed is sent to the sender (Alice) through networking.
    d. If the parity is different between the two
        i. The number of errors in the pass is incremented
        ii. The block is split into 2 and the parity of each is checked with the parity on Alice's side. This continues recursively until the exact error location in the block is identified.
        iii. The error location in the block is mapped to the error location in the key. The value of the bit is not flipped on Bob's side as it is flipped on Alice's side.
5. The last bit of each block is removed from the key so that the parity information sent in step 4 would be useless for any eavesdropper who might have been able to record them. Moreover, a bit is removed for each group whose parity has been sent over the network (this applies to the smaller blocks whose parities are sent on the network in step 4.d.ii)
6. The current block size is incremented with the block size increment.

Steps 1 to 6 are repeated over so that the key would be passed over the error correction phase with different block size comparisons. When the key iterates 20 passes without having any discrepancy between the parities at Alice and Bob's side, it is assumed that the key is now the same with both of them.

At this stage, the error correction phase computes the error rate to check whether it is in the acceptable limits. This is done by applying the following steps:

1. The total number of errors is calculated which is a count of the errors encountered at the different stages of the error correction phase.
2. An estimate of errors that hasn't been encountered is made by interpolation. These errors were undiscovered because they have been removed when deleting the last bit of each block; thus, they were removed but not counted as errors by the algorithm so we need to interpolate them.

3. The total number of errors is calculated as the sum of the counts computed at steps 1 and 2.
4. The error rate is computed to be the total errors divided by the number of bits in the initial key.

If the error rate is too high, then the algorithm aborts because of the hazard of an eavesdropper and the first phase of the BB84 protocol (the creation of the keys) should be re-applied. The error rate induced by the several techniques of eavesdropping would be higher than 11% and thus 11% has been taken the threshold of the error rate. [Gisin et al. 2002].

As can been noted from the previous steps, the work done on Bob's side is nearly the same as the one done at Alice's side with some modifications. Thus, in our implementation we created a base class that contains all the common functionality and inherited it in the Sender and Receiver classes.

5.3 Enhanced Error Correction algorithm considerations

The objective of this research is to make enhancements in the error correction phase of the BB84 protocol to make it more efficient and thus to decrease the number of bits lost during this phase. This objective is to be realized through the use of memory between the rounds of error correction phase where a round would identify the locations it found errors in to help the next round to be more focused. This would decrease the number of rounds used to reach the final shared string and increase the shared string size.

The proposed enhancement to achieve this goal is to keep an array of the history of each bit in the shared key and use it before making the key comparisons using parity in each round. This history attribute for each bit would keep the number of times the bit has participated in blocks that had parity checks with the other communicating party before. Thus, this history attribute could be queried to check whether the bit has passed a parity check in a block before and if it has passed such parity checks, the history attribute could tell the number of times it had passed such parity checks. Keeping this history attribute is beneficial as each round could make such a query before choosing to check on the identified bit again. Thus, at the start of each round the enhanced algorithm would query the history table and identify the bits that have been thoroughly checked in the previous rounds. If a bit has been proven correct using parity check in different blocks where the parity has been the same with both communicating parties, then this bit doesn't get included in the new round. Doing this decreases the number of bits participating in the parity check of the round, and thus less number of rounds would be needed; moreover, this would mean that the number of bits removed at the end of the round decreases as the exclusion of some bits from the round would decrease the number of blocks compared in that round. Decreasing the number of rounds means decreasing the number of bits removed as the last bit at each block is removed.

Using the history table information to exclude bits from entering next rounds would not affect the correctness of the algorithm and its ability to achieve correct results. A bit is excluded from entering next rounds only when it is thoroughly checked in previous rounds and has passed at least 5 correct parity checks in different blocks. Since the maximum error rate would be 11% there is an upper limit on the number of errors and those parity checks are enough for accepting the bit value as correct. This was more apparent in the experiments described in chapter 6 where the enhanced algorithm got correct results as the standard one but it creates longer keys in less time.

5.4 Enhanced Algorithm Implementation

As with the implementation of the standard algorithm, the enhanced algorithm is built up of 2 modules which are the “Sender” (Alice) and the “Receiver” (Bob) where each of them does the part needed from the respective user during the error correction phase. Detailed description of the modules for the Sender and the receiver follows:

5.4.1 Sender (Alice) module

When running the algorithm implementation as Alice, the error correction phase implementation receives an initial key value which represents the key that Alice has. The key is represented by an array of bits. The following tasks are applied:

1. The bits history array is set to zeros.
2. Alice creates a comparison key from the key present at the time. The comparison key is the same as the initial key at the first round; however, it is computed in the other rounds as will be shown later in step 10.
3. Alice computes the number of blocks that it would split the key to. The number of blocks is calculated to be equal the number of bits in the comparison key divided by the current block size. [The current block size is equal to the initial block size at the start. The initial block size is one of the parameters that are set for the error correction phase].
4. Alice creates a random permutation of numbers from 1 till the size of the comparison key that she has. This permutation is used later to shuffle the locations of the bit values in the key array.
5. Alice sends the permutation created in step 4 to Bob. This is done through TCP over the network so the “Receiver” (Bob) module could be running on a different machine. By receiving the permutation, Bob and Alice would have the same permutation and could use it to shuffle the locations in the key array if they want

to. This permutation would be later used to shuffle the order by which the bits are compared.

6. Alice partitions the comparison key into blocks each of the size of the current block size (the current block size is the initially the initial block size that is a parameter to the error correction phase). The bit positions of the each block are taken from the key and are computed using the permutation created in step 3.The number of blocks has been computed in step 3 noted earlier.
7. For each block the following is applied:
    a. The parity of the block is calculated.
    b. The parity computed is sent to the receiver (Bob) through networking.
    c. The parity of the corresponding bit at Bob's side is received.
    d. If the parity is different between the two
        i. The number of errors in the pass is incremented
        ii. The block is split into 2 and the parity of each is checked with the parity on Bob's side. This continues recursively until the exact error location in the block is identified.
        iii. The error location in the block is mapped to the error location in the key and its value is flipped (if it were 0, it would be changed to 1 and vice versa).
        iv. The bits history array is reset to zero for all the bits participating in the block to indicate that the bits of this block has participated in an erroneous block.
    e. If the parity is the same between the two communicating parties:
        i. The bits history array is incremented for all the bits participating in the block to indicate that the bits of this block have participated in a correct block.
8. The last bit of each block is removed from the key so that the parity information sent in step 7 would be useless for any eavesdropper who might have been able to record them. Moreover, a bit is removed for each group whose parity has been sent over the network (this applies to the smaller blocks whose parities are sent on the network in step 7.d.ii)

9. The current block size is incremented with the block size increment.
10. The comparison key for the next round is computed. The comparison key is computed to contain all the bits that have bit history values less than 5. Having a bit history value less than 5 means that the bit has participated in less than 5 successful block parity comparisons before. Excluding the bit from the comparison means that we consider that it is enough that a bit has passed 5 comparisons to consider it as the same at the sender's and receiver's side.

Steps 2 to 10 are repeated over so that the key would be passed over the error correction phase with different block size comparisons. When the key iterates 20 passes without having any discrepancy between the parities at Alice and Bob's side, it is assumed that the key is now the same with both of them.

At this stage, the error correction phase computes the error rate to check whether it is in the acceptable limits. This is done by applying the following steps:

1. The total number of errors is calculated which is a count of the errors encountered at the different stages of the error correction phase.
2. An estimate of errors that hasn't been encountered is made by interpolation. These errors were undiscovered because they have been removed when deleting the last bit of each block; thus, they were removed but not counted as errors by the algorithm so we need to interpolate them.
3. The total number of errors is calculated as the sum of the counts computed at steps 1 and 2.
4. The error rate is computed to be the total errors divided by the number of bits in the initial key.

If the error rate is too high, then the algorithm aborts because of the hazard of an eavesdropper and the first phase of the BB84 protocol (the creation of the keys) should be re-applied. The error rate induced by the several techniques of eavesdropping would be higher than 11% and thus 11% has been taken the threshold of the error rate. [Gisin et al. 2002].

5.4.2 Receiver (Bob) module

When running the algorithm implementation as Bob, the error correction phase implementation receives an initial key value which represents the key that Bob has. The key is represented by an array of bits. The following tasks are applied:

1. The bits history array is set to zeros.
2. Bob creates a comparison key from the key present at the time. The comparison key is the same as the initial key at the first round; however, it is computed in the other rounds as will be shown later in step 10.
3. Bob computes the number of blocks that it would split the key to. The number of blocks is calculated to be equal the number of bits in the comparison key divided by the current block size. [The current block size is equal to the initial block size at the start. The initial block size is one of the parameters that are set for the error correction phase].
4. Bob receives the permutation created by Alice. This is done through TCP over the network. By receiving the permutation, Bob will have the same permutation Alice has and could use it to shuffle the locations in the key array if they want to. This permutation would be later used to shuffle the order by which the bits are compared.
5. Bob partitions the key into blocks each of the size of the current block size (the current block size is the initially the initial block size that is a parameter to the error correction phase). The bit positions of the each block are taken from the key and are computed using the permutation received in step 4.The number of blocks has been computed in step 3 noted earlier. The blocks at Bob and Alice side are the same as they applied the same algorithm to create them.
6. For each block the following is applied:
    a. The parity of the block is calculated.
    b. The parity of the corresponding bit at Alice's side is received.
    c. The parity computed is sent to the sender (Alice) through networking.
    d. If the parity is different between the two
        i. The number of errors in the pass is incremented

        ii. The block is split into 2 and the parity of each is checked with the parity on Alice's side. This continues recursively until the exact error location in the block is identified.
        iii. The error location in the block is mapped to the error location in the key. The value of the bit is not flipped on Bob's side as it is flipped on Alice's side.
        iv. The bits history array is reset to zero for all the bits participating in the block to indicate that the bits of this blocks has participated in an erroneous block.
    e. If the parity is the same between the two communicating parties:
        i. The bits history array is incremented for all the bits participating in the block to indicate that the bits of this block have participated in a correct block.

7. The last bit of each block is removed from the key so that the parity information sent in step 6 would be useless for any eavesdropper who might have been able to record them. Moreover, a bit is removed for each group whose parity has been sent over the network (this applies to the smaller block whose parities are sent on the network in step 7.d.ii)
8. The current block size is incremented with the block size increment.
9. The comparison key for the next round is computed. The comparison key is computed to contain all the bits that have bit history values less than 5. Having a bit history value less than 5 means that the bit has participated in less than 5 successful block parity comparisons before. Excluding the bit from the comparison means that we consider that it is enough that a bit has passed 5 comparisons to consider it as the same at the sender's and receiver's side.

Steps 2 to 9 are repeated over so that the key would be passed over the error correction phase with different block size comparisons. When the key iterates 20 passes without having any discrepancy between the parities at Alice and Bob's side, it is assumed that the key is now the same with both of them.

At this stage, the error correction phase computes the error rate to check whether it is in the acceptable limits. This is done by applying the following steps:

1. The total number of errors is calculated which is a count of the errors encountered at the different stages of the error correction phase.
2. An estimate of errors that hasn't been encountered is made by interpolation. These errors were undiscovered because they have been removed when deleting the last bit of each block; thus, they were removed but not counted as errors by the algorithm so we need to interpolate them.
3. The total number of errors is calculated as the sum of the counts computed at steps 1 and 2.
4. The error rate is computed to be the total errors divided by the number of bits in the initial key.

If the error rate is too high, then the algorithm aborts because of the hazard of an eavesdropper and the first phase of the BB84 protocol (the creation of the keys) should be re-applied. The error rate induced by the several techniques of eavesdropping would be higher than 11% and thus 11% has been taken the threshold of the error rate. [Gisin et al. 2002].

As can been noted from the previous steps, the work done on Bob's side is nearly the same as the one done at Alice's side with some modifications (the same case as the standard algorithm). Thus, in our implementation we created a base class that contains all the common functionality and inherited it in the Sender and Receiver classes.

Chapter 6

Test results & Assessment

Chapter 6: Test results & Assessment

6.1 Experimentation Scenarios

A systematic approach was applied to make the experimentation and extract the results on both the standard and the enhanced implementations of the error correction phase. The aim was to be able to assess the performance of both implementations under different criteria that might affect them. The criteria that are of effect on the error correction phase are the following:

1. The number of bits initially present as a shared key between the sender and the receiver (Alice and Bob) at the start of the error correction phase.
2. The number of discrepancies between Alice's and Bob's bit strings. This is the number of errors that should be captured by the error correction phase.
3. The initial (basic) block size. The algorithm splits the shared key into number of blocks, each with the size of the initial block size. These are the blocks that are used for parity comparison between Alice and Bob (the details are in chapter 5).
4. The block size increment. This is the increment added to the block size at the end of each round in the error correction phase. Thus, a round would use a block size which is bigger than the block size used at the previous round (the difference between the 2 block sizes is the block size increment).

The algorithm performance on a run is calculated based upon a set of counters that monitor the key strings established between Alice and Bob at the end of the error correction phase. Moreover, there are counters that calculate the time needed to reach the shared key and the discrepancies initially present between Alice's and Bob's keys. The full set of counters and their significance is detailed in Table 6-1.

Table 6-1 Counters for the error correction Phase

Counter	Description
Number of Passes	The number of rounds needed for the error correction phase to settle on the final key between Alice and Bob
Total Operations	The number of comparisons (parity checks) between Alice's and Bob's bits
Key Length	The length of the final key that is shared between Alice and Bob at the end of the error correction phase
Evaluation	This is a calculated value which is equal to the Key Length divided by the total operations. It indicates the relation between the key length with the time needed to reach it. This value needs to be maximized for best performance
Errors Found	This is the number of discrepancies that the error correction phase was able to find between Alice's and Bob's initial key strings
Interpolated Errors	This is the number of errors interpolated based upon the number of errors that were actually calculated
Total Errors	This is the sum of errors found and interpolated errors. This value should be the same as the number of discrepancies that were initially present between Alice's and Bob's keys
Error Rate	This is the calculated error rate. It is calculated to be equal to the total errors divided by the number of bits initially present at the start of the error correction phase. If the error rate is too high, then the transmission is aborted.

At the start of each scenario, the error correction phase implementation takes the IP of the other part in order to be able to contact it during the running of the phase. Both the Sender and receiver module take as input a key string which is the shared key between them. The errors are induced at this phase for testing the performance of the algorithm at different key sizes and different number of errors. After that the algorithm runs till it reaches a shared key at the end. Moreover, it calculates all the needed counters for the run; most importantly it calculates the error rate to check if it's acceptable or not. Additional counters are saved for each round of the running of the algorithm such as the number of errors discovered at each round; however, these are round-specific and they are summed to be the total errors found so they weren't included in the comparison tables.

6.2 Standard Algorithm experimental results (2000 bits)

6.2.1 80 discrepancies:

The first set of experiments applied on the standard algorithm were done using initial key size of 2000 bits and an error rate of 4% which is equivalent to 80 errors. We started up with those values because they were the values described during the experimentation of Bennet and Brassard in 1992 [Bennet et al. 1992]. Applying those values with an initial block size of 5 and having a variable block size increment gave us the results in Table 6-2

Table 6-2 Standard Algorithm (2000 Bits – 80 Errors) with basic block size =5

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
5	5	13	820	1180	351	1.43902439	75	10.82234071	85.822341	4.291117035
5	10	14	723	1277	349	1.766251729	73	8.99880448	81.998804	4.099940224
5	15	16	678	1322	350	1.949852507	72	8.267173191	80.267173	4.01335866
5	20	14	633	1367	357	2.159557662	69	8.109548109	77.109548	3.855477405
5	25	15	631	1369	376	2.169572108	74	8.909344015	82.909344	4.145
5	30	16	620	1380	379	2.225806452	72	7.633981865	79.633982	4.247946608
5	35	17	630	1370	398	2.174603175	74	8.560371998	82.560372	4.1280186
5	40	16	618	1382	401	2.236245955	74	8.235404804	82.235405	4.11177024
5	50	16	600	1400	397	2.333333333	72	8.362372536	80.362373	4.01
5	60	18	607	1393	406	2.294892916	71	7.483042096	78.483042	3.92
5	70	18	575	1425	378	2.47826087	69	8.284535308	77.284535	3.86
5	80	21	639	1361	435	2.129890454	69	5.552581534	74.552582	3.73
5	90	31	673	1327	448	1.971768202	72	6.537829534	78.53783	3.93
5	100	19	563	1437	374	2.552397869	68	9.21084105	77.210841	3.86
5	110	23	612	1388	410	2.267973856	69	7.685441204	76.685441	3.83

6.2.2 160 discrepancies

A second set of experiments were made on the standard algorithm with an initial key of 2000 bits but having 160 errors (8%). The same set of experiments applied with 80 errors is repeated with 160 errors. However, we included only the results of using the basic block size at 5 as the conclusions are the same as the ones obtained when having the number of errors 80. The results are tabled in Table 6-3 and plotted as shown in Fig. 6-2.

Table 6-3 Standard Algorithm (2000 Bits – 160 Errors) with basic block size =5

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
5	10	17	1023	977	668	0.955034213	133	20.1144255	153.1144255	7.655721275
5	20	25	986	1014	677	1.028397566	121	13.00691227	134.0069123	6.700345613
5	30	37	1099	901	792	0.819836215	126	11.23760789	137.2376079	6.861880395
5	40	42	1099	901	798	0.819836215	127	12.46022502	139.460225	6.973011251
5	50	40	1107	893	829	0.806684734	126	11.43821454	137.4382145	6.871910727
5	60	41	1058	942	788	0.890359168	123	13.61079945	136.6107995	6.830539973
5	70	43	1086	914	814	0.841620626	124	12.82846635	136.8284664	6.841423318
5	80	53	1146	854	847	0.745200698	123	11.06986019	134.0698602	6.70349301
5	90	54	1172	828	897	0.706484642	127	9.838432808	136.8384328	6.84192164
5	100	46	1153	847	886	0.734605377	129	12.8702212	141.8702212	7.09351106
5	110	55	1155	845	881	0.731601732	131	14.67097427	145.6709743	7.283548713
5	120	64	1161	839	876	0.722652885	126	12.43118062	138.4311806	6.921559031
5	130	50	1162	838	889	0.721170396	129	11.99941486	140.9994149	7.049970743
5	140	58	1186	814	904	0.686340641	127	9.656704764	136.6567048	6.832835238
5	150	68	1232	768	931	0.623376623	127	9.283777199	136.2837772	6.81418886
5	160	64	1188	812	907	0.683501684	127	11.0253032	138.0253032	6.90126516
5	170	61	1196	804	933	0.672240803	129	11.07246697	140.072467	7.003623349
5	180	65	1175	825	910	0.70212766	128	11.56317819	139.5631782	6.97815891
5	190	70	1185	815	911	0.687763713	127	11.71902184	138.7190218	6.935951092
5	200	58	1201	799	916	0.665278934	130	10.9336661	140.9336661	7.046683305

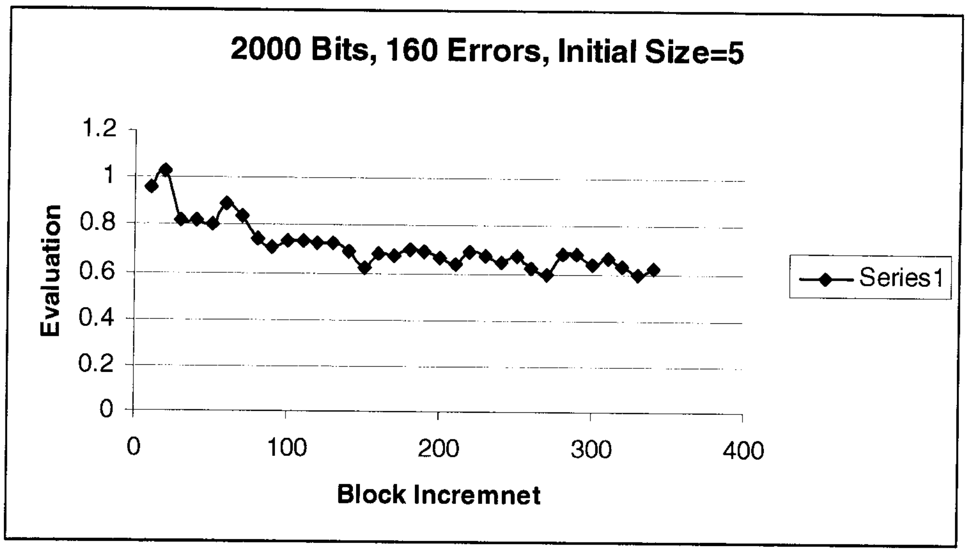


Figure 6-2 Standard Algorithm (2000 Bits – 160 Errors) with basic block size =5

We notice from the graph that when the error rate increased (from 4% to 8%), the number of bits removed has increased and the key length, reached at the end of the error correction phase, became shorter and thus the evaluation value has decreased. When there are160 errors, the maximum value reached for the evaluation is 1.02; however, when there were 80 errors, the evaluation reached 2.5 (Actually, this observation is predictable because when the errors increase, we expect the key reached at the end to be smaller).

6.3 Enhanced Algorithm experimental results (2000 bits)

The experiments that have been tested on the standard algorithm are replayed again on the enhanced algorithm to be able to assess the performance of the enhanced algorithm under the same conditions the standard algorithm has been tested.

6.3.1 80 discrepancies:

The first set of experiments were applied to the standard algorithm were done using initial key size as 2000 bits and the error rate at 4% which means that there were 80 errors. This was the first test for the enhanced algorithm too. Applying those values with an initial block size of 5 and having a variable block size increment gave us the results in Table 6-4

Table 6-4 Enhanced Algorithm (2000 Bits – 80 Errors) with basic block size =5

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
5	5	15	1079	1076	156	0.997219648	68	17.05315822	85.05315822	4.252657911
5	10	15	891	1270	162	1.425364759	68	17.05315822	85.05315822	4.252657911
5	15	16	815	1358	174	1.666257669	70	17.5547217	87.5547217	4.377736085
5	20	15	737	1407	145	1.909090909	62	15.54846779	77.54846779	3.87742339
5	25	15	700	1442	143	2.06	60	15.04690432	75.04690432	3.752345216
5	30	16	703	1464	168	2.082503556	68	17.05315822	85.05315822	4.252657911
5	35	16	680	1483	164	2.180882353	68	17.05315822	85.05315822	4.252657911
5	40	14	653	1494	148	2.287901991	60	15.04690432	75.04690432	3.752345216
5	45	15	658	1504	163	2.285714286	70	17.5547217	87.5547217	4.377736085
5	50	17	631	1514	146	2.399366086	62	15.54846779	77.54846779	3.87742339
5	55	17	626	1520	147	2.428115016	60	15.04690432	75.04690432	3.752345216
5	60	17	623	1527	151	2.451043339	62	15.54846779	77.54846779	3.87742339
5	65	16	618	1531	150	2.477346278	62	15.54846779	77.54846779	3.87742339
5	70	20	612	1535	148	2.508169935	62	15.54846779	77.54846779	3.87742339
5	75	19	607	1541	149	2.538714992	64	16.05003127	80.05003127	4.002501563
5	80	20	600	1541	142	2.568333333	60	15.04690432	75.04690432	3.752345216
5	85	22	618	1543	162	2.496763754	66	16.55159475	82.55159475	4.127579737
5	90	18	601	1547	149	2.574043261	64	16.05003127	80.05003127	4.002501563
5	95	18	622	1551	174	2.493569132	70	17.5547217	87.5547217	4.377736085
5	100	21	616	1548	165	2.512987013	72	18.05628518	90.05628518	4.502814259

Plotting the values of the evaluation versus block increment having the other values as constant gives us Fig. 6-3

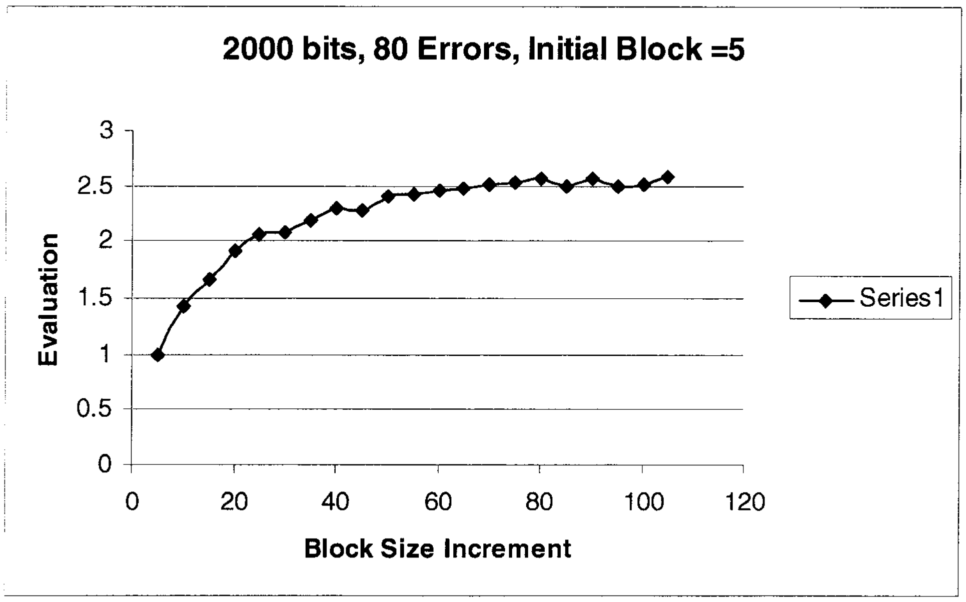


Figure 6-3 Enhanced Algorithm (2000 Bits – 80 Errors) with basic block size =5

The same parameters were used but with changing the basic block size value to become 10 in some experiments and 20 in others. The tables containing the results for those experiments and the figures plotted using those values are presented in the appendix in table A-3 and Fig A-3 (for the case of using basic block size at 10) and table A-4 and Fig A-4 (for the case of using basic block size at 20).

It is apparent from comparing those figures with the corresponding figures using the standard algorithm that there hasn't been much improvement due to using the new algorithm when starting with a block size of 5. However, increasing the initial block size to become 10 and 20 showed a real difference from the standard algorithm. The standard algorithm reached a maximum of 2.5 for the evaluation at 10 or 20 initial block size. Using the enhanced algorithm, the algorithm reaches till 4.5 using 10 as the initial block size and reaches 8 when using 20 as the initial block size which is a great improvement over the standard algorithm.

6.4 Comparison Graphs for 2000 bits

The results and graphs obtained in the previous sections are concentrating on having an initial key of 2000 bits with 80 discrepancies between the sender's and receiver's bits. Figures 6-4, 6-5 and 6-6 combine the graphs of both the standard and enhanced algorithm for the different values of the initial block size. Those graphs are used for the comparison of the results of applying both the standard and the enhanced algorithms.

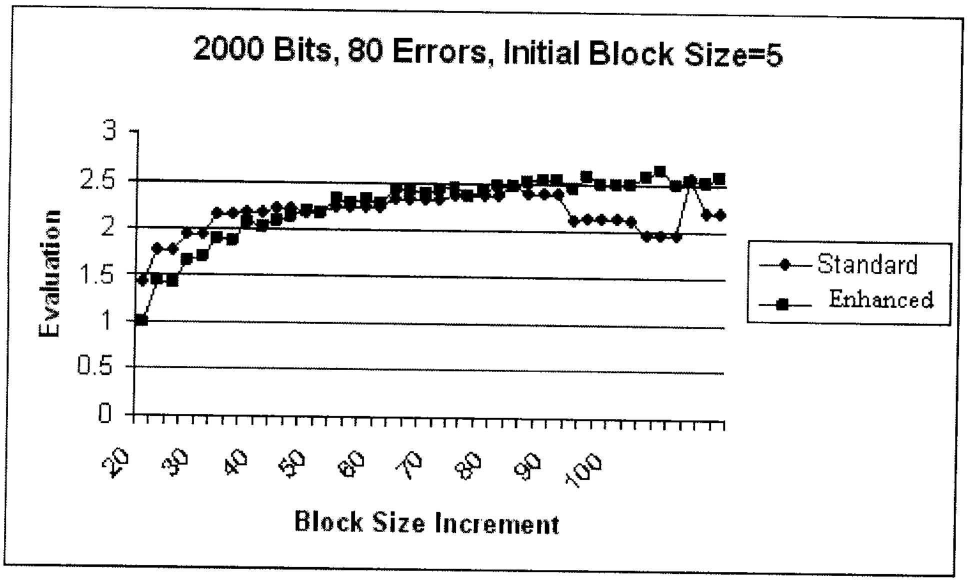


Figure 6-4 Standard vs. Enhanced (2000 Bits – 80 Errors) Basic block size =5

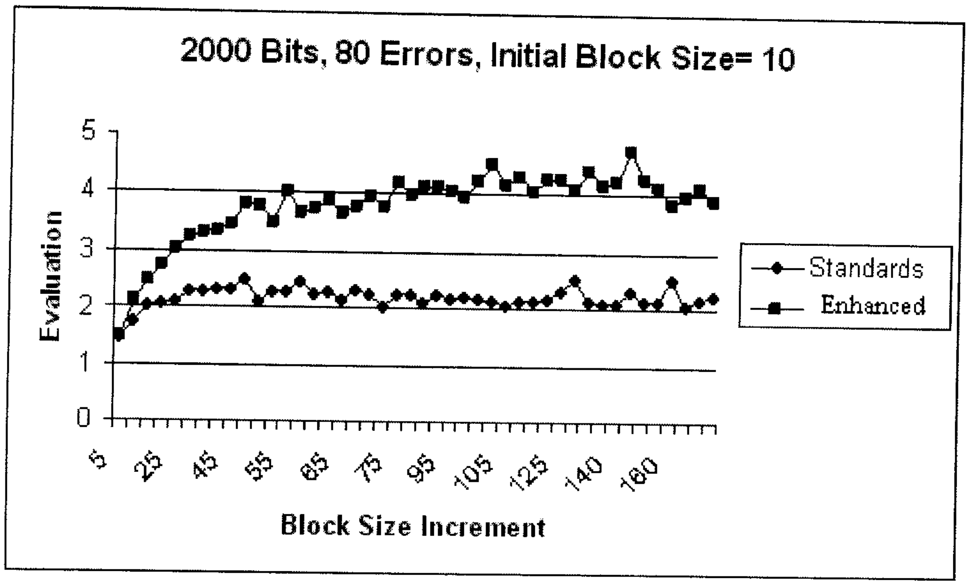


Figure 6-5 Standard vs. Enhanced (2000 Bits – 80 Errors) Basic block size =10

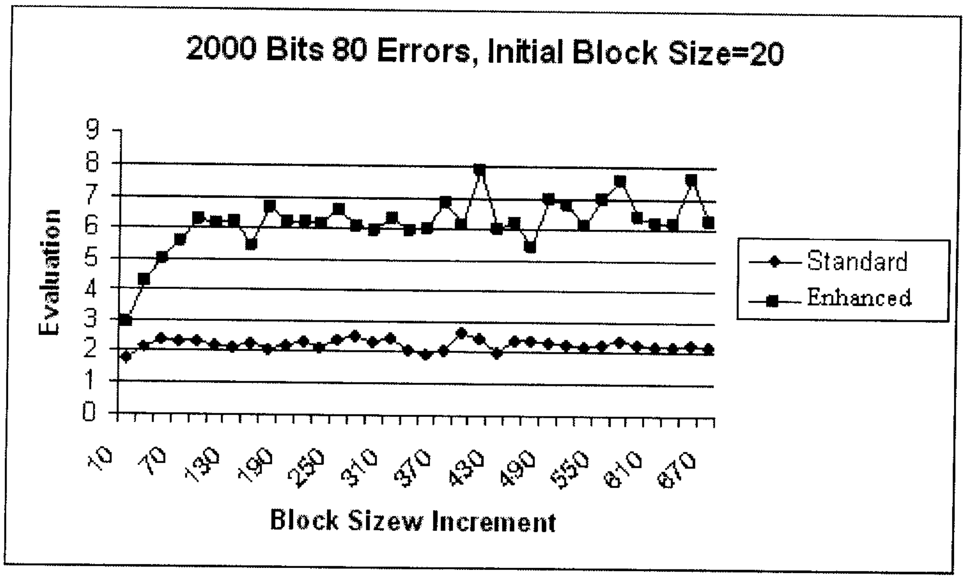


Figure 6-6 Standard vs. Enhanced (2000 Bits – 80 Errors) Basic block size =20

These comparison graphs make it clear that at the initial block size of 5, there is no significant difference between the results obtained by the standard and the enhanced algorithms; however, when increasing the basic block size to 10 or 20, the gap between the standard and the enhanced algorithm performances became wider and the superiority of the enhanced algorithm became apparent. Using basic block size of 10, the standard algorithm reached a maximum evaluation value of 2.5; however, the enhanced algorithm gave results at the range of 4-5 which is much better of course. The same result is concluded when using the basic block size at value 20. Using basic block size at 20, the standard algorithm works in the range of 2.5 maximum (just as it did when the basic block size was 5 and 10); however, the enhanced algorithm has reached even a better performance result at the range of 6-8 which is better than the results obtained when the basic block size was set to 10. A complete mathematical model for the results of the algorithm and a quantitative measure of the performance differences between the standard and the enhanced algorithms is present at chapter 7 "Algorithm Behavioral Mathematical Model".

6.5 Standard Algorithm experimental results (5000 bits)

The experiments were repeated but using a 5000-bit initial key string between both parties rather than the 2000-bit one used earlier. The same experiments were applied with 100 discrepancies between the sender's and receiver's initial key strings. The experiments were re-applied with 200 discrepancies between the bit strings. The results of those experiments are noted in the following sections.

6.5.1 100 discrepancies

Table 6-5 represents the result of applying the experiments on the standard algorithm with 100 discrepancies between the sender and the receiver with basic block size set to 10. Figure 6-7 is the figure obtained from the result value when using basic block size of 10. The same parameters were used but with changing the basic block size value to become 20 in some experiments and 30 in others. The tables containing the results for those experiments and the figures plotted using those values are presented in the appendix in table A-5 and Fig A-5 (for the case of using basic block size at 20) and table A-6 and Fig A-6 (for the case of using basic block size at 30).

Table 6-5 Standard Algorithm (5000 Bits – 100 Errors) with basic block size =10

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
10	5	13	1699	3301	443	1.942907593	97	9.689542693	106.6895427	2.133790854
10	10	13	1407	3593	448	2.55366027	97	9.399259681	106.3992597	2.127985194
10	20	13	1178	3822	474	3.244482173	100	10.72916108	110.7291611	2.214583222
10	30	14	1077	3923	478	3.642525534	98	9.942087394	107.9420874	2.158841748
10	40	14	992	4008	464	4.040322581	97	10.05352272	107.0535227	2.141070454
10	50	14	978	4022	494	4.112474438	98	9.338807285	107.3388073	2.146776146
10	60	14	969	4031	519	4.15995872	99	8.807000647	107.8070006	2.156140013
10	70	14	918	4082	490	4.446623094	98	10.29973406	108.2997341	2.165994681
10	80	19	986	4014	557	4.070993915	98	7.478176506	105.4781765	2.10956353
10	90	15	906	4094	509	4.518763797	96	9.439546529	105.4395465	2.108790931
10	100	15	917	4083	532	4.452562704	97	8.454485651	105.4544857	2.109089713
10	110	16	918	4082	536	4.446623094	96	8.299112724	104.2991127	2.085982254
10	120	16	911	4089	540	4.488474204	97	8.38173421	105.3817342	2.107634684
10	130	17	915	4085	546	4.464480874	96	8.0746881	104.0746881	2.081493762
10	140	16	878	4122	519	4.69476082	96	9.214196863	105.2141969	2.104283937
10	150	15	900	4100	550	4.555555556	97	8.396540709	105.3965407	2.107930814

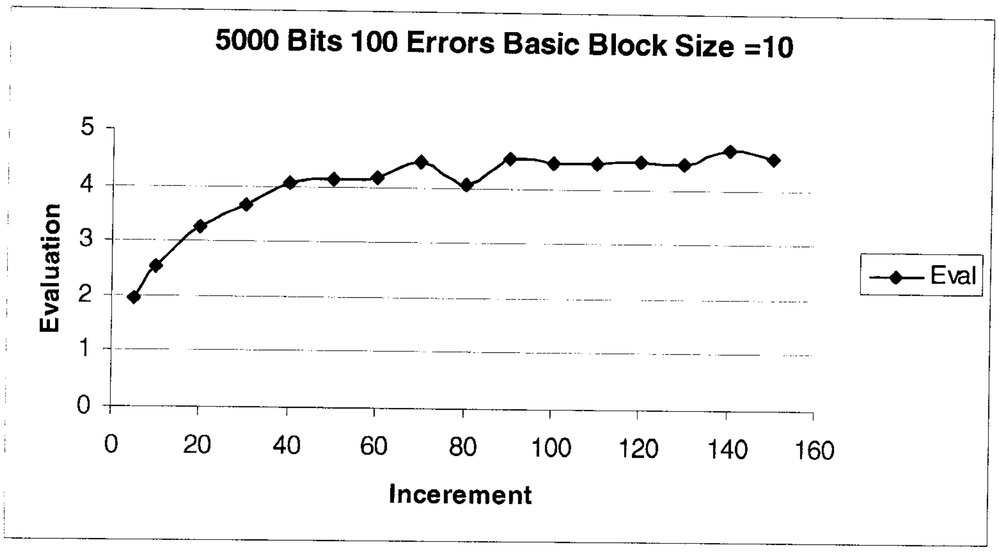


Figure 6-7 Standard Algorithm (5000 Bits – 100 Errors) with basic block size =10

We notice from the previous figures that the initial block size slightly affects the result obtained. As can be seen, the evaluation has a maximum of 4.9 and the basic block size value doesn't have an effect on it. When the basic block size was set to10, 20 or 30 nearly the same results are obtained. This observation is the same as the observation made when a 2000-bit key string was used in the previous sections.

6.5.2 200 discrepancies

Table 6-6 represents the result of applying the experiments on the standard algorithm with 200 discrepancies between the sender and the receiver with basic block size set to 10. Figure 6-8 is the figure obtained from the result value when using basic block size of 10. The same parameters were used but with changing the basic block size value to become 20 in some experiments and 30 in others. The tables containing the results for those experiments and the figures plotted using those values are presented in the appendix in table A-7 and Fig A-7 (for the case of using basic block size at 20) and table A-8 and Fig A-8 (for the case of using basic block size at 30).

Table 6-6 Standard Algorithm (5000 Bits – 200 Errors) with basic block size =10

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
10	5	15	2105	2895	859	1.375296912	186	22.81446488	208.8144649	4.176289298
10	10	15	1863	3137	920	1.683843264	190	21.29912971	211.2991297	4.225982594
10	20	15	1608	3392	913	2.109452736	183	21.77195016	204.7719502	4.095439003
10	30	17	1562	3438	969	2.201024328	189	24.34720828	213.3472083	4.266944166
10	40	18	1563	3437	1030	2.198976328	183	18.4106364	201.4106364	4.028212728
10	50	20	1545	3455	1051	2.236245955	183	18.94069574	201.9406957	4.038813915
10	60	19	1455	3545	998	2.436426117	179	21.23259806	200.2325981	4.004651961
10	70	22	1507	3493	1057	2.317850033	177	16.79910418	193.7991042	3.875982084
10	80	32	1676	3324	1205	1.983293556	186	16.38900033	202.3890003	4.047780007
10	90	28	1558	3442	1117	2.209242619	183	19.17486354	202.1748635	4.043497271
10	100	40	1666	3334	1185	2.00120048	181	16.08614686	197.0861469	3.941722937
10	110	40	1679	3321	1209	1.977963073	185	18.1863215	203.1863215	4.06372643
10	120	37	1626	3374	1176	2.07503075	183	17.46471829	200.4647183	4.009294366
10	130	39	1658	3342	1205	2.015681544	182	17.24965623	199.2496562	3.984993125
10	140	38	1623	3377	1180	2.080714726	185	19.76314081	204.7631408	4.095262816
10	150	46	1729	3271	1256	1.891844997	181	14.8129663	195.8129663	3.916259326

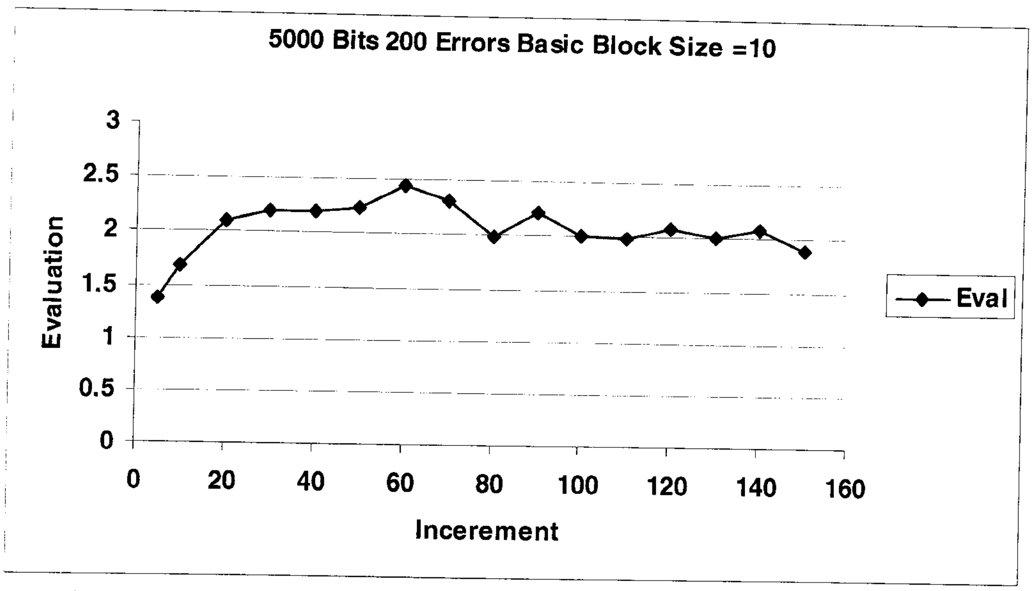


Figure 6-8 Standard Algorithm (5000 Bits – 200 Errors) with basic block size =10

The same observations that were noticed when using 2000 bits are still observed with the increase of the number of bits in the initial key string. We notice that when the error rate increased (from 2% to 4%), the number of bits removed has increased and the key length, reached at the end of the error correction phase, became shorter and thus the evaluation value has decreased. When there are 200 errors, the maximum value reached for the evaluation is 2.5; however, when there were 100 errors, the evaluation reached 4.9

6.6 Enhanced Algorithm experimental results (5000 bits)

The same experiments applied to the standard algorithm were applied to the enhanced algorithm with 100 discrepancies between the sender's and receiver's initial key strings, then with 200 discrepancies between the bit strings. The results of those experiments are noted in the following sections.

6.6.1 100 discrepancies

Tables 6-7 represents the result of applying the experiments on the enhanced algorithm with 100 discrepancies between the sender and the receiver with basic block size set to 10. Figure 6-9 is the figure obtained from the result value when using basic block size of 10. The same parameters were used but with changing the basic block size value to become 20 in some experiments and 30 in others. The tables containing the results for those experiments and the figures plotted using those values are presented in the appendix in table A-9 and Fig A-9 (for the case of using basic block size at 20) and table A-10 and Fig A-10 (for the case of using basic block size at 30).

Table 6-7 Enhanced Algorithm (5000 Bits – 100 Errors) with basic block size =10

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
10	5	12	1826	3173	173	1.737677985	50	7.041511454	57.04151145	1.140830229
10	10	12	1443	3554	175	2.462924463	50	7.224380743	57.22438074	1.144487615
10	20	13	1150	3849	183	3.346956522	50	6.883211081	56.88321108	1.137664222
10	30	13	1018	3979	179	3.908644401	49	7.157853449	56.15785345	1.123157069
10	40	13	956	4041	190	4.226987448	50	6.424071494	56.42407149	1.12848143
10	50	13	898	4101	181	4.566815145	49	6.354772479	55.35477248	1.10709545
10	60	14	871	4129	187	4.740528129	50	7.052847137	57.05284714	1.141056943
10	70	14	859	4141	197	4.820721769	50	6.371744706	56.37174471	1.127434894
10	80	15	836	4163	194	4.979665072	50	6.310513844	56.31051384	1.126210277
10	90	15	837	4162	207	4.972520908	50	6.000308833	56.00030883	1.120006177
10	100	16	790	4210	174	5.329113924	50	7.324737491	57.32473749	1.14649475
10	110	16	798	4201	192	5.264411028	50	6.651385659	56.65138566	1.133027713
10	120	16	783	4217	185	5.385696041	49	6.655301291	55.65530129	1.113106026
10	130	17	790	4208	199	5.326582278	49	6.412023306	55.41202331	1.108240466
10	140	16	774	4226	191	5.45994832	50	7.03041153	57.03041153	1.140608231
10	150	16	802	4196	221	5.2319202	49	5.757744048	54.75774405	1.095154881

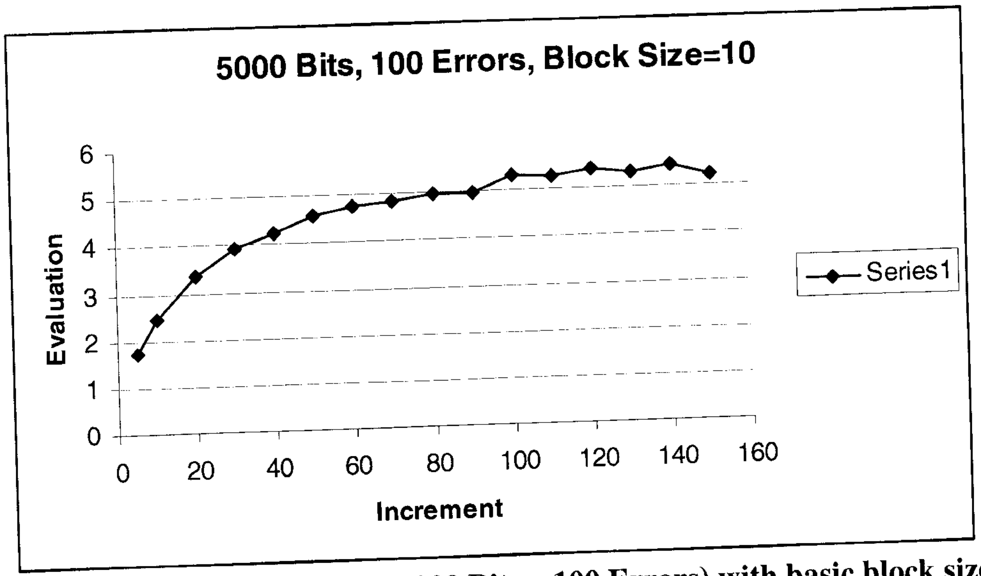


Figure 6-9 Enhanced Algorithm (5000 Bits – 100 Errors) with basic block size =10

It can be noticed from the previous graphs that as the basic block size increases, better results are reached. When the basic block size was 10, the maximum evaluation reached was around 5.5; however, when the basic block size was 20, the maximum evaluation was 7.7. When the basic block size was 30, the evaluation reached a maximum of 8.6. Thus, the enhanced algorithm reaches better results when the basic block size is incremented (contrary to the standard algorithm that doesn't get affected by the value of the basic block size).

6.6.2 200 discrepancies

The same experiments were re-applied to the enhanced algorithm and the graphs conveyed the same results previously noted.

6.7 Comparison Graphs for 5000 bits

The results and graphs obtained in the section are concentrating on having an initial key of 5000 bits with 100 discrepancies between the sender's and receiver's bits. Figures 6-10, 6-11 and 6-12 combine the graphs of both the standard and enhanced algorithm for the different values of the initial block size. Those figures are used for the comparison of the results of applying both the standard and the enhanced algorithms.

The next figures are similar to figures 6-4, 6-5 and 6-6 that were used to compare the performance of the standard and the enhanced algorithm for 2000 bits initial key.

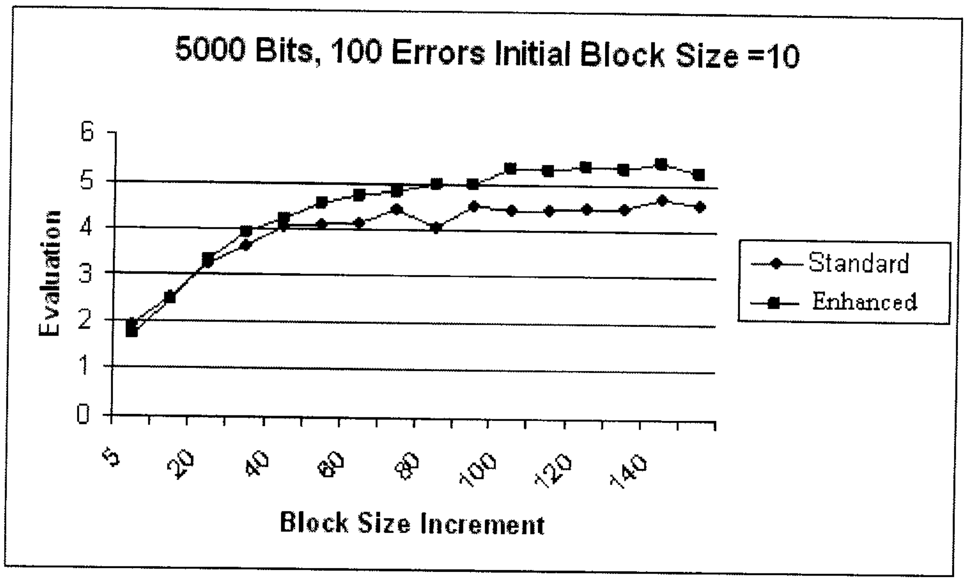


Figure 6-10 Standard vs. Enhanced (5000 Bits – 100 Errors) Basic block size =10

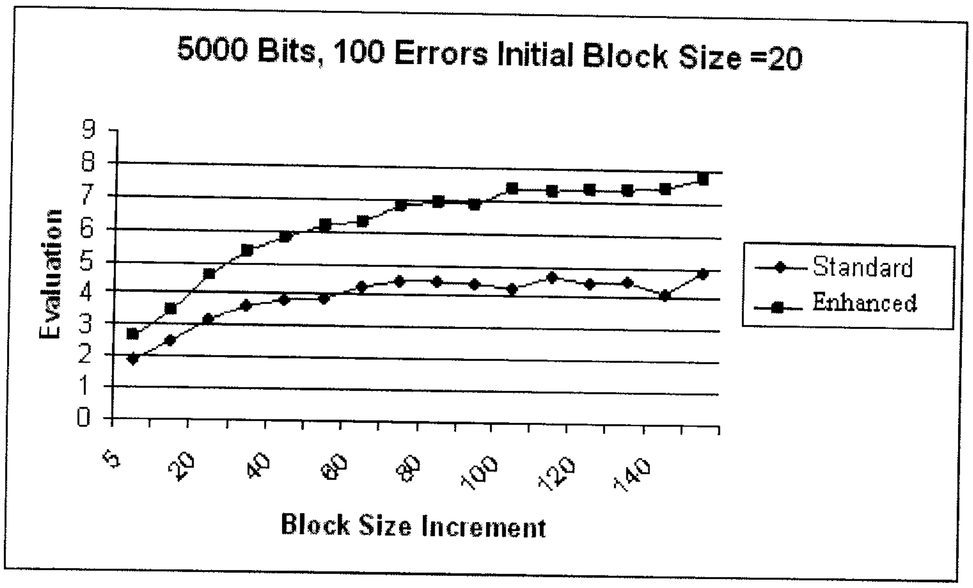


Figure 6-11 Standard vs. Enhanced (5000 Bits – 100 Errors) Basic block size =20

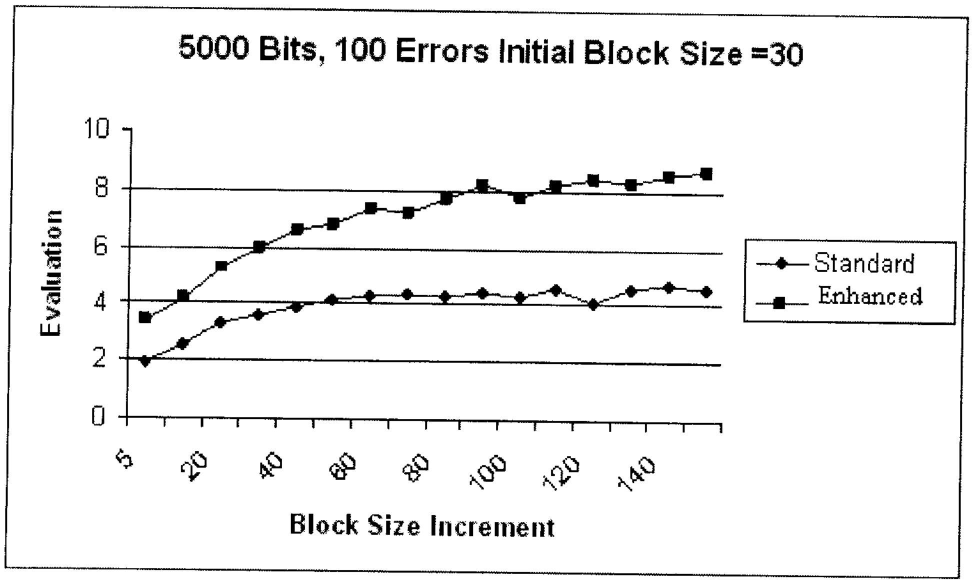


Figure 6-12 Standard vs. Enhanced (5000 Bits – 100 Errors) Basic block size =30

These comparison graphs make it clear that at the initial block size of 10, there is some difference between the results obtained by the standard and the enhanced algorithm; however, when increasing the basic block size to 20 or 30, the gap between the standard and the enhanced algorithm performances became wider and the superiority of the enhanced algorithm became apparent. Using basic block size of 10, the standard algorithm reached a maximum evaluation value of 4.7; however, the enhanced algorithm gave results that reached 5.4. Using basic block size at 20, the standard algorithm works in the range of 4.8 maximum; however, the enhanced algorithm has reached even a better performance result at the range of 6-7.8 which is better than the results obtained when the basic block size was set to 10. Using basic block size of 30, the standard algorithm stayed in the same range of evaluation values, but the enhanced algorithm was able to reach 8.6. A complete mathematical model for the results of the algorithm and a quantitative measure of the performance differences between the standard and the enhanced algorithms is present at chapter 7 “Algorithm Behavioral Mathematical Model”.

6.8 Error Rate Effect

6.8.1 Effect of increasing error rate

During the experimentation on the standard and the enhanced algorithm, some experiments were made where the error rate was made to be 25% and above. The purpose of doing such experiments with more than 25% discrepancies between the sender's and receiver's initial bit strings is to monitor the performance of the error correction phase algorithm in very noisy environments or under heavy eavesdropping attack [Eavesdropping on quantum key distribution induces errors which are detectable later through the error rate].

Applying those experiments revealed a drawback that must be overcome. The problem was the following: when the algorithm (the standard or the enhanced one) was applied to such bit strings with high error rates, the algorithm would return calculating an error rate that is a low value (much lower than the actual error rate value). Moreover, the key created at the end of the error correction phase on the sender's side is different than the key created at the receiver's size. Both the above results are catastrophic on the result of the error correction phase due to the following reasons. Having the error correction phase estimating a wrong value for the error rate means that it might accept transmissions that it should have refused. For example, if the error rate was actually 20% then the transmission should be aborted; however, if the error correction phase calculated the error rate to be 9% only, then the transmission would be considered safe and no abortion will occur. Having different keys at the sender's and the receiver's side at the end of the error correction phase is a failure of this phase in achieving its goal. The key that results at the end of the error correction phase should be the key that would be later used for encryption/decryption between the sender and the receiver; thus, having discrepancies in such a key would mean that the sender and the receiver would encrypt/decrypt with different keys and all their communications would fail.

Tables 6-8, 6-9 and 6-10 show the results of applying the standard algorithm with an initial key size of 5000 and 1000 errors between the sender's and receiver's bits. Thus, there are 5000 bit-string and 20% error rate between the sender and the receiver.

Only the first two rows of each of tables 6-8, 6-9 and 6-10 got the same keys between the sender and the receiver. The other rows in the table denote experiments where there were discrepancies between the keys at the sender and receiver side.

Table 6-8 Standard Algorithm (5000 Bits – 1000 Errors) with basic block size =10

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
10	5	31	4056	944	2912	0.232741617	563	73.90838997	636.90839	12.7381678
10	10	64	4191	809	3124	0.193032689	525	50.07388681	575.0738868	11.50147774
10	20	103	4092	908	3131	0.221896383	462	34.42243044	496.4224304	9.928448609
10	30	149	4356	644	3434	0.147842057	483	39.07704322	522.0770432	10.44154086
10	40	173	4281	719	3352	0.167951413	448	30.87568324	478.8756832	9.577513665
10	50	167	4200	800	3345	0.19047619	437	30.38521809	467.3852181	9.347704362
10	60	178	4167	833	3310	0.199904008	431	31.07262217	462.0726222	9.241452443
10	70	182	4160	840	3312	0.201923077	429	32.26685709	461.2668571	9.225337142
10	80	184	4105	895	3291	0.218026797	416	28.33671306	444.3367131	8.886734261
10	90	181	4088	912	3255	0.223091977	399	21.40972709	420.4097271	8.408194542
10	100	180	4001	999	3200	0.249687578	394	24.30192791	418.3019279	8.366038558
10	110	176	3997	1003	3215	0.250938204	404	28.11197131	432.1119713	8.642239426
10	120	194	4031	969	3209	0.240387001	397	26.01450921	423.0145092	8.460290184
10	130	193	3978	1022	3184	0.256913022	397	28.41779286	425.4177929	8.508355857
10	140	191	4021	979	3219	0.243471773	405	30.73590519	435.7359052	8.714718104
10	150	196	3951	1049	3147	0.265502404	389	26.44330576	415.4433058	8.308866115

Table 6-9 Standard Algorithm (5000 Bits – 1000 Errors) with basic block size =20

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
20	5	34	4076	924	2878	0.226692836	544	65.46302824	609.4630282	12.18926056
20	10	77	4328	672	3227	0.155268022	539	54.35394905	593.3539491	11.86707898
20	20	141	4444	556	3410	0.125112511	490	36.35875388	526.3587539	10.52717508
20	30	157	4363	637	3416	0.146000458	469	31.97513501	500.975135	10.0195027
20	40	155	4268	732	3382	0.171508903	458	36.03218713	494.0321871	9.880643743
20	50	166	4190	810	3312	0.193317422	429	28.13103502	457.131035	9.1426207
20	60	179	4174	826	3312	0.197891711	426	30.13093595	456.130936	9.122618719
20	70	179	4173	827	3323	0.198178768	422	26.62692925	448.6269292	8.972538585
20	80	175	4024	976	3196	0.242544732	389	19.81772921	408.8177292	8.176354584
20	90	187	4096	904	3267	0.220703125	412	27.59730812	439.5973081	8.791946162
20	100	186	4099	901	3270	0.21980971	418	33.25412191	451.2541219	9.025082438
20	110	182	4009	991	3222	0.247193814	403	26.76301887	429.7630189	8.595260377
20	120	184	4031	969	3231	0.240387001	402	28.90739683	430.9073968	8.618147937
20	130	195	3955	1045	3148	0.264222503	388	24.86286959	412.8628696	8.257257392
20	140	178	4034	966	3257	0.239464551	399	25.33950464	424.3395046	8.486790093
20	150	189	3939	1061	3156	0.269357705	392	27.31233071	419.3123307	8.386246614

Table 6-10 Standard Algorithm (5000 Bits – 1000 Errors) with basic block size =30

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
30	5	32	4153	847	3000	0.203948953	569	70.09186777	639.0918678	12.78183736
30	10	70	4284	716	3199	0.16713352	531	51.05647105	582.056471	11.64112942
30	20	132	4464	536	3450	0.120071685	499	35.46597815	534.4659782	10.68931956
30	30	155	4342	658	3393	0.151543068	465	32.44818428	497.4481843	9.948963686
30	40	164	4288	712	3399	0.166044776	462	35.32792119	497.3279212	9.946558424
30	50	164	4186	814	3329	0.194457716	437	29.70453768	466.7045377	9.334090754
30	60	169	4146	854	3287	0.205981669	416	24.29150347	440.2915035	8.805830069
30	70	177	4079	921	3239	0.225790635	412	27.2112711	439.2112711	8.784225422
30	80	191	4137	863	3275	0.20860527	415	28.73350267	443.7335027	8.874670053
30	90	196	4121	879	3271	0.213297743	419	32.81343998	451.81344	9.0362688
30	100	199	4133	867	3278	0.209774982	410	26.9385553	436.9385553	8.738771106
30	110	187	4011	989	3208	0.246571927	403	27.79802444	430.7980244	8.615960489
30	120	182	4041	959	3248	0.237317496	406	28.37927271	434.3792727	8.687585454
30	130	202	3949	1051	3115	0.266143327	387	25.16057295	412.160573	8.243211459
30	140	185	4027	973	3230	0.241619071	398	26.55284794	424.5528479	8.491056959
30	150	207	3974	1026	3147	0.258178158	387	25.57991394	412.5799139	8.251598279

We notice from the graphs that nearly all of the results are wrong in the sense that they produce different keys at the sender and the receiver side. Moreover, we notice that the calculated error rate is around 8% or 9% which is much less than the actual error rate (20%). Miscalculating the error rate by that amount is dangerous as it could hide the presence of an eavesdropper. For example, if the threshold was 11% (any error rate above that would mean that the communication is insecure and should be aborted), then all the above experiments would pass as they are below the threshold though their actual error rate is 20%. This problem appears more significant when we have 2000 errors (error rate =40%) as all the results are erroneous and the algorithm assumes that they are right and have low error rates.

The same effect appears in the enhanced algorithm too as it is an enhancement of the standard algorithm.

6.8.2 Cause and Solution

The above problem occurs because of the way the blocks are checked between the sender's and the receiver's side. In BB84, the sender creates a block, checks its parity and sends the parity to the receiver. The receiver applies the same scenario and both sides check the local and the other side's parity. If the parities are the same, the BB84 protocol assumes that the blocks are the same depending on the fact that later blocks would verify that in later rounds. This way of comparison works fine with a low number of discrepancies between the 2 keys at both sides; however, when the error rate increases, using the parity to check the blocks doesn't give the correct results as there are multiple errors in any block. This is the cause of the problem noted in the previous section which is the erroneous results of the protocol when the error rate increases.

The solution to this problem is to have an estimate of the error rate prior to starting the parity checks on the keys. Having such an estimate before-hand would help the error correction phase to abort early before it starts the parity checks phase; moreover, it avoids the problem described in the previous section [Xu 2002].

Thus, the error correction phase algorithm would be modified to add a part that would calculate such an estimate. The modifications are made to the algorithm implementation on both sides (the sender and the receiver). The modification would need to apply the following steps:

1. Create a sample block at both the sender's and the receiver's sides.
2. Compare the values of each bit in the block between the sender and the receiver.
3. Calculate the error rate of the block.

The value calculated in the previous steps would be a representative sampling rate that would be used to estimate the error rate.

6.8.2.1 Alice (Sender)

The following steps are added at the start of the error correction phase at the sender's side to be able to calculate an estimate of the error rate:

1. Alice creates a random array of locations in the initial key. This array contains random locations in the initial key that would be used to create a sample to check the error rate with.
2. Alice sends the array of locations to the receiver (Bob).
3. Alice sends the value of each element in the sample to Bob and receives Bob's value.
4. Alice compares her value with the value received from Bob. If the values are different then the number of errors is incremented.
5. Alice calculates the sample error rate (Number of errors / Sample block Size). If the error rate is above the threshold, then the transmission is aborted.
6. If the sample error rate is acceptable, then Alice removes all the bits that has participated in the sample and starts the algorithm for the error correction phase.

6.8.2.2 Bob (Receiver)

The following steps are added at the start of the error correction phase at the receiver's side to be able to calculate an estimate of the error rate:

1. Bob receives the array of locations sent by Alice.
2. Bob receives the value of each element in the sample from Alice and sends his value for the same element.
3. Bob compares her value with the value received from Alice. If the values are different then the number of errors is incremented.
4. Bob calculates the sample error rate (Number of errors / Sample block Size). If the error rate is above the threshold, then the transmission is aborted.
5. If the sample error rate is acceptable, then Bob removes all the bits that has participated in the sample and starts the algorithm for the error correction phase.

Chapter 7

Algorithm Behavioral Mathematical Model

Chapter 7: Algorithm Behavioral Mathematical Model

In order to assess the effect of the enhanced algorithm on the performance of the error correction phase, there is a need to quantify the experimental results and have a mathematical model for the system. There is also a need to note the effect of changing the parameters of the algorithm such as the initial block size and the block size increment on both the standard and the enhanced algorithms. The work done in this direction is explained in this chapter.

7.1 Result Graphs

A set of experiments were applied to both the standard and the enhanced algorithm with changing values of basic block size and block size increment. The graphs of the results of the experiments are detailed in this section. The first set of the experiments were applied using 5000 bits with 100 and 200 errors (error rate = 2% and 4% respectively).

7.1.1 Standard algorithm

Figures 7.1 and 7.2 represent the results of applying the standard algorithm on a bit string of 5000 bits with 100 and 200 errors respectively. In both those cases, the basic block size was kept constant at 10 and the block size increment was varying each time to be able to identify its effect on the final algorithm result.

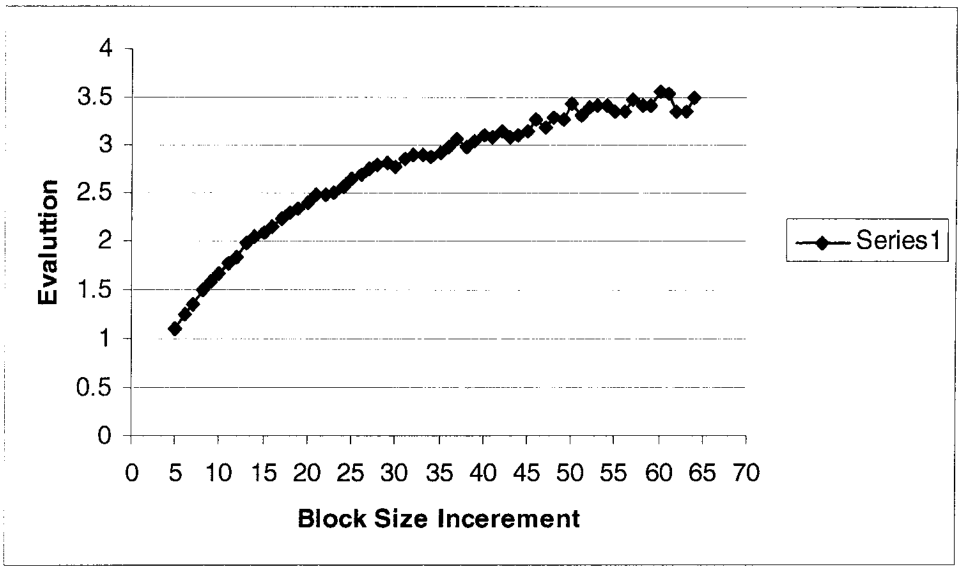


Figure 7-1 Standard Algorithm (5000 Bits – 100 Errors) with basic block size =10

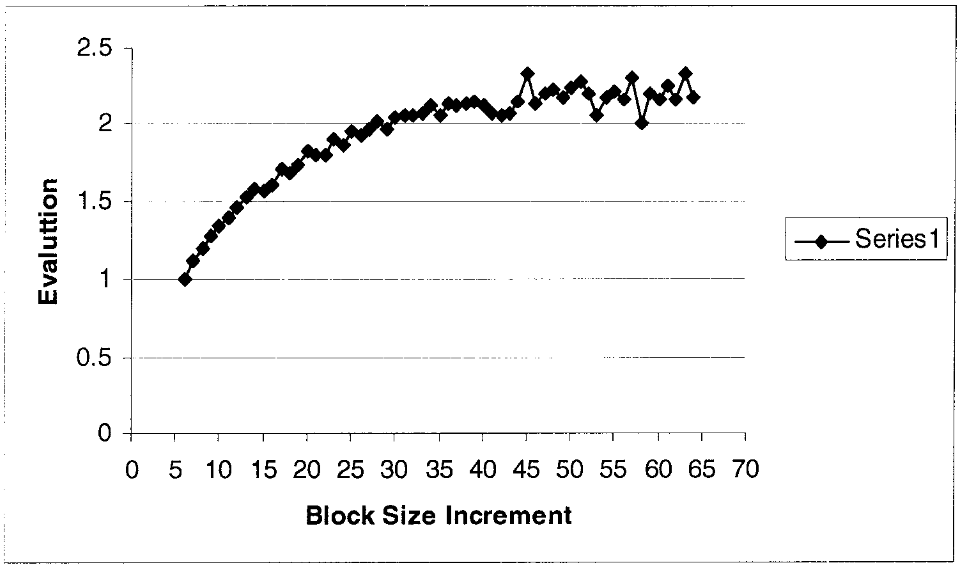


Figure 7-2 Standard Algorithm (5000 Bits – 200 Errors) with basic block size =10

It is apparent from the graphs that the block size increment has an effect on the final outcome of the algorithm. As the block size increment increases, the evaluation result gets higher until it reaches a maximum limit and it saturates. When there were 100 errors, the saturation was at around 3.56 and when there were 200 errors, the saturation

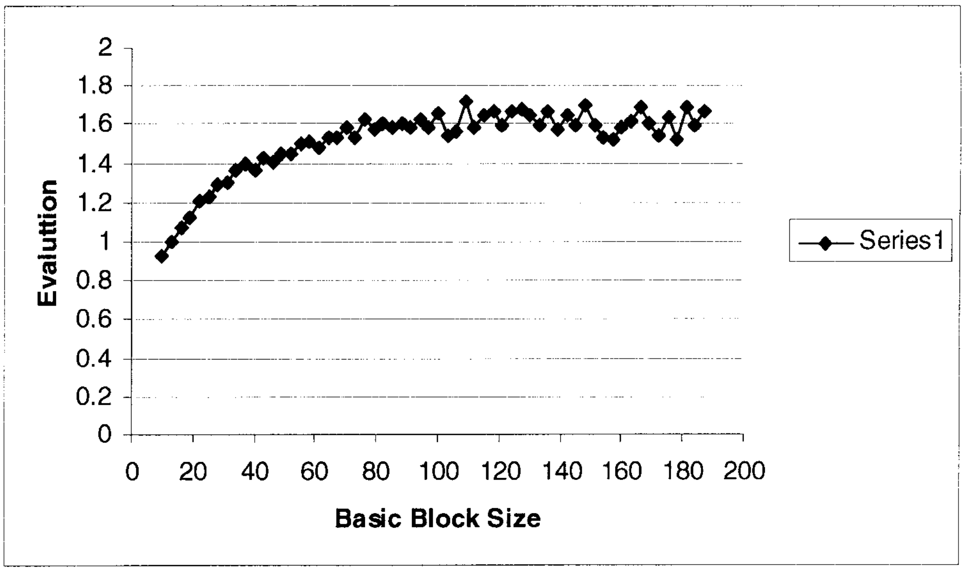


Figure 7-4 Standard Algorithm (5000 Bits – 200 Errors) with block size inc. =5

It is apparent from the figures that the basic block size has an effect on the final outcome of the algorithm. As the basic block size increases, the evaluation result gets higher until it reaches a maximum limit and it saturates. When there were 100 errors, the saturation was at around 3.15 and when there were 200 errors, the saturation was at 1.715.

7.1.2 Enhanced Algorithm

The same set of experiments and results were done on the enhanced algorithm. Figures 7.5 and 7.6 represent the results of applying the enhanced algorithm on a bit string of 5000 bits with 100 and 200 errors respectively. In both those cases, the basic block size was kept constant at 10 and the block size increment was varying each time to be able to identify its effect on the final algorithm result.

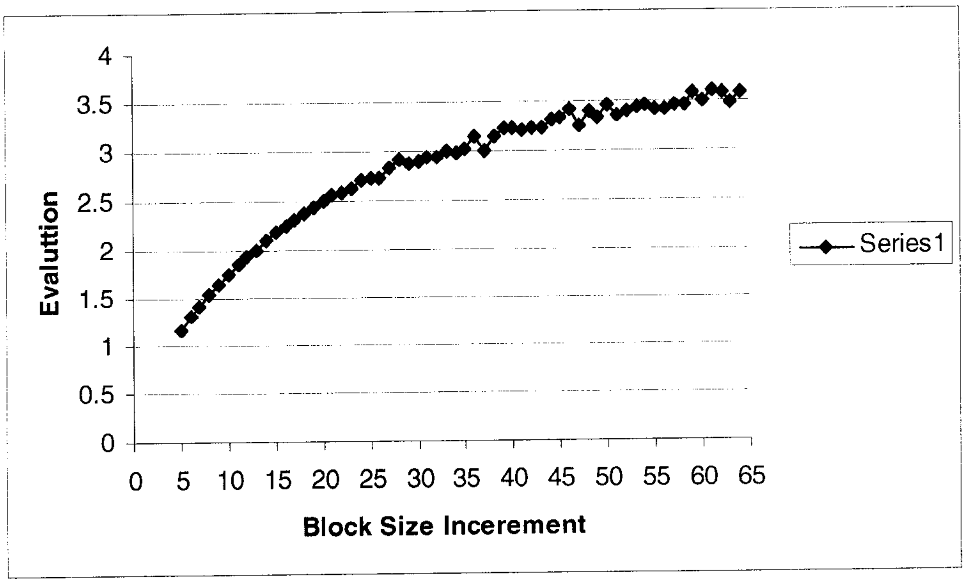


Figure 7-5 Enhanced Algorithm (5000 Bits – 100 Errors) with basic block size =10

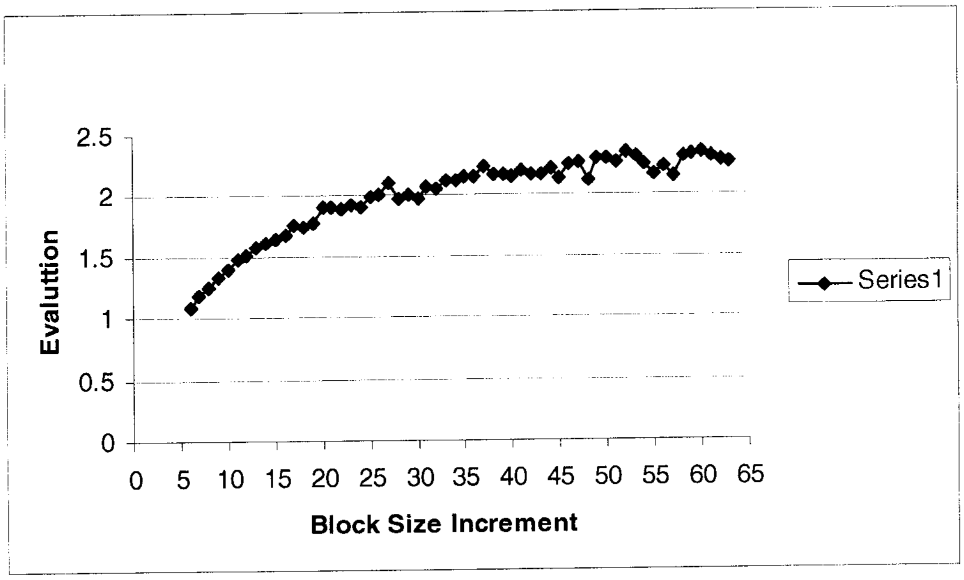


Figure 7-6 Enhanced Algorithm (5000 Bits – 200 Errors) with basic block size =10

The same observation noted in the standard algorithm is noted here too where the block size increment affects the result of the algorithm.

Figures 7.7 and 7.8 represent the results of applying the enhanced algorithm on a bit string of 5000 bits with 100 and 200 errors respectively. In both those cases, the block size increment was kept constant at 5 and the basic block size was varying each time to be able to identify its effect on the final algorithm result.

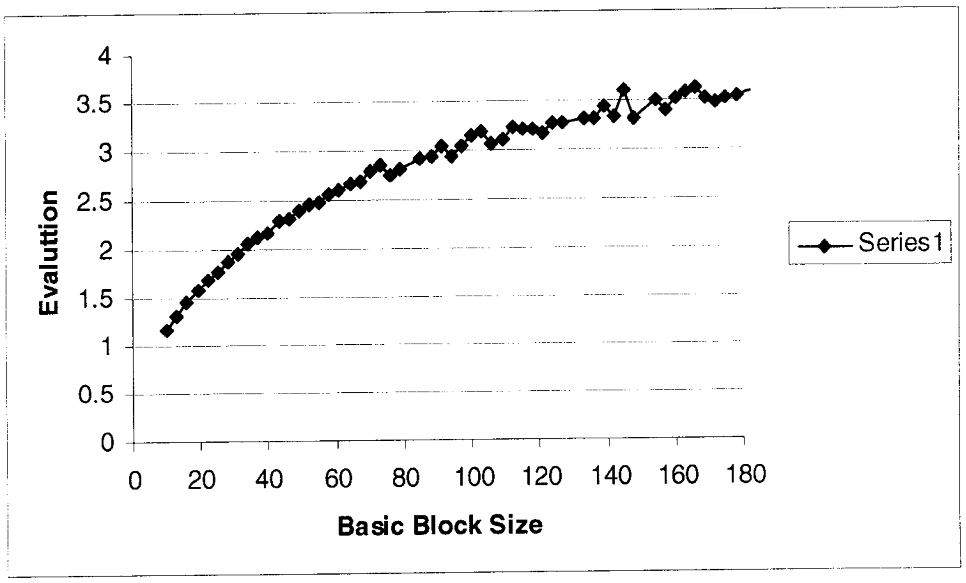


Figure 7-7 Enhanced Algorithm (5000 Bits – 100 Errors) with block size inc. =5

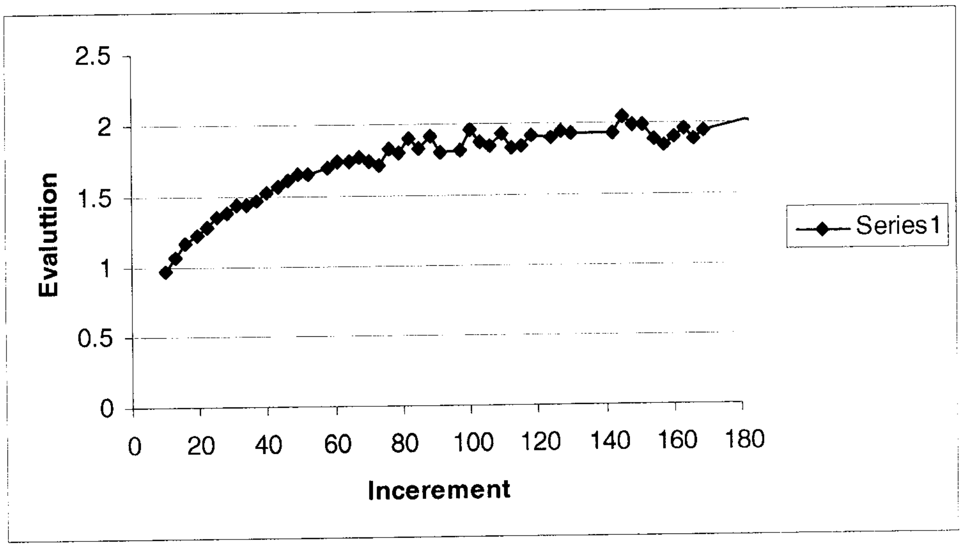


Figure 7-8 Enhanced Algorithm (5000 Bits – 200 Errors) with block size inc. =5

The same observation noted in the standard algorithm is noted here too where the basic block size affects the result of the algorithm.

7.2 Mathematical Model

All the above graphs exhibit the same behavior which is mainly trying to reach a value and saturate at it. This is apparent in all the figures of chapter 6 as the evaluation always move on increasing until it reaches a maximum value and then the evaluation would saturate at that value. Trying to model such a behavior mathematically yields the function $y = a(1-e^{-bx})$ where x is the input, y is the output. **a** and **b** are constants where a is the maximum value that would be reached and b indicates the speed of moving towards the saturation value. [For the previous graphs of this chapter, y would be the "evaluation" values, and x would be the "basic block size" or the "block size increment"].

Using the $y = a(1-e^{-bx})$ as a model of the behavior would mean that we should calculate the **a** and **b** values for the standard and enhanced algorithm under the different basic block size and block size increment values. Calculating the **a** and **b** values for such cases would show the effect of changing each value and the effect of using the enhanced algorithm versus the standard algorithm. Those values could then be used as a quantitative way to measure the effect of the changes made on the results obtained.

First, some mathematical analysis should be made to be able to have a clear path on how to calculate the **a** and **b** values from figures like the ones present in this chapter (figures 7-1 to 7-8). Calculating the value of a is obvious as it is the saturation value that the evaluation value tries to reach so it is easy to note the value from the graph; however, calculating the value of b needs some extra calculations to be made.

In the equation $Y = a(1-e^{-bX})$

Y is the evaluation, X is the input and **a** is calculated directly from the graph

Thus, the only unknown is **b**:

Since $Y = a(1-e^{-bX})$

Then $\frac{Y}{a} = 1-e^{-bX}$

Thus $-bX = \ln(1-\frac{Y}{a})$ [Equation 7.1]

Let $Z = \ln(1 - \frac{Y}{a})$

For any point (i) $Z_i = -bX_i$

The error between the value returned for the equation and the actual value is E. We are trying to reduce this error and our optimum is to have it being equal to zero. Using square error

$$E = \sum_{i=1}^{n} (Z_i(actual) - Z_i(calculated))^2$$

Our aim is to have $\frac{\partial E}{\partial b} = 0$

Thus, $\frac{\partial E}{\partial b} = \frac{\partial}{\partial b}[\sum_{i=1}^{n} (Z_i(actual) - Z_i(calculated))^2]$

$$= \frac{\partial}{\partial b}[\sum_{i=1}^{n} (Z_i(actual) + bX_i)^2]$$

$$= \frac{\partial}{\partial b}[\sum_{i=1}^{n} (Z_i^2 + 2bZ_iX_i + b^2X_i^2)\ ]$$

$$= \frac{\partial}{\partial b}[2b\sum_{i=1}^{n} Z_iX_i + b^2\sum_{i=1}^{n} X_i^2\ ]$$

$$= 2\sum_{i=1}^{n} Z_iX_i + 2b\ \sum_{i=1}^{n} X_i^2\ ]$$

Since $\frac{\partial E}{\partial b} = 0$

Then, $2\sum_{i=1}^{n} Z_iX_i + 2b\ \sum_{i=1}^{n} X_i^2\ ] = 0$

$$b = -(\frac{\sum_{i=1}^{n} Z_iX_i}{\sum_{i=1}^{n} X\ _i^2})$$

Thus, $$b = -\left(\frac{\sum_{i=1}^{n} \ln(1-\frac{Y_i}{a}\ X_i}{\sum_{i=1}^{n} X_i^2}\right)$$ [Equation 7.2]

Thus we calculate **b** using the equation above.

7.3 Applying Mathematical Model on results

The mathematical model described above creates a model for the experiments and a quantitative way to show the effect of making those changes on the results obtained. The experiments are modeled by the equation Since $Y = a(1 - e^{-bX})$ where **a** and **b** are constants that will be calculated for each experiment. Y is the evaluation function, and X is the variable that would change during the experimentation (the basic block size or the block size increment).

The results of applying this model using the standard and enhanced algorithm are detailed as follows. All the experiments done at this point use 10000-bits key strings in order to have a large range of error values and thus a broader range of results obtained.

7.3.1 Standard Algorithm

As stated earlier, all experiments are done with 10000-bits initial key string. It is noted from equation 7.1 that $\ln(1 - \frac{Y}{a}) = -bX$, which means that if $\ln(1 - \frac{Y}{a})$ is drawn in relation to X, the result should be a straight line whose slope is –b. This graph is presented for each experiment later on.

The results of applying the standard algorithm on a 10000-bits key string with 50 errors and a basic block size of 10 are shown in Fig. 7.9

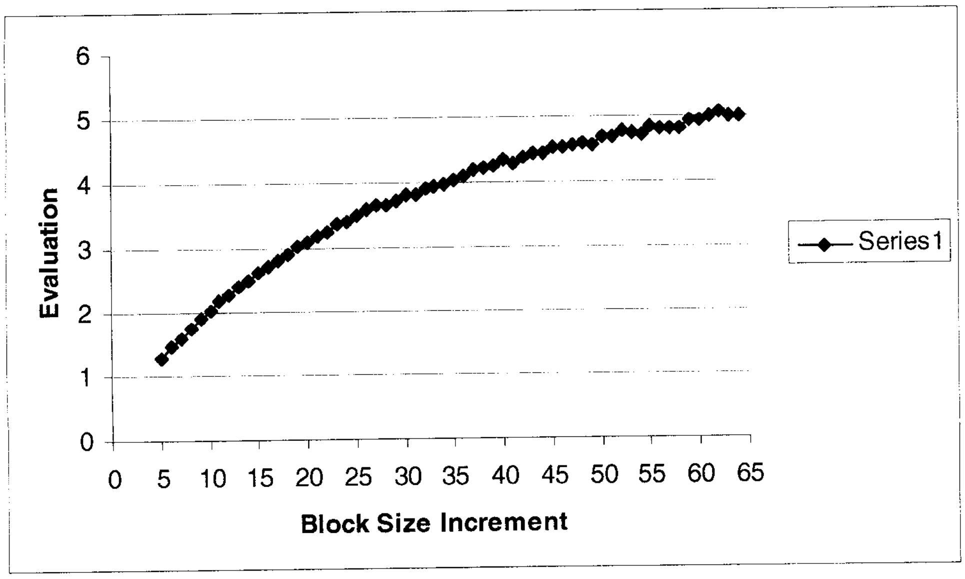


Figure 7-9 Standard Algorithm (10000 Bits – 50 Errors) with basic block size =10

After that the value of **a** is estimated, which is 5.06 in this case, and the graph of block size increment versus $\ln(1-\frac{\text{Evaluation}}{a})$ is plotted as seen in Fig. 7-10

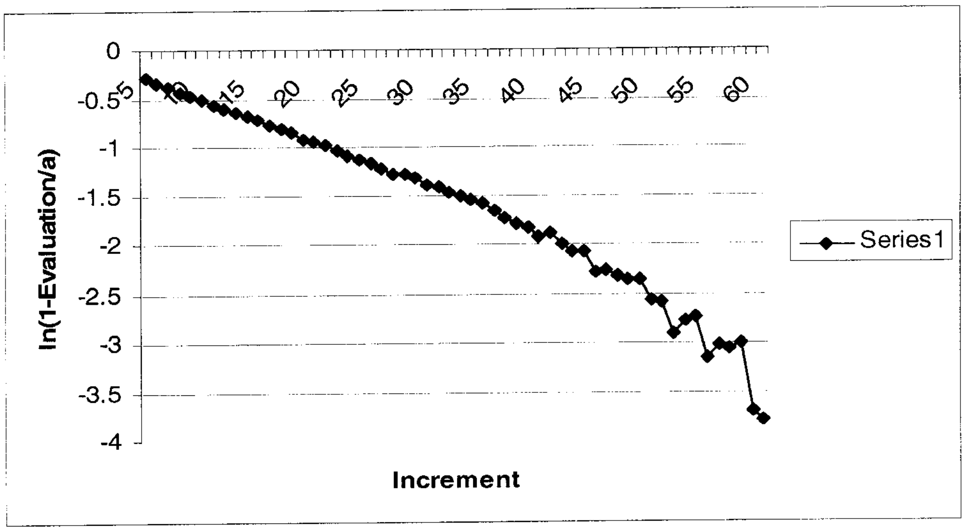


Figure 7-10 Standard Algorithm (10000 Bits – 50 Errors)

Block Size Inc. vs. ln (1-Eval/a)

The value of **b** is calculated according to equation 7.2 and it was 0.001173759 in this case.

Applying this same scenario with a 10000 bits-key strings with 100 errors showed the results in figures 7.11 and 7.12.

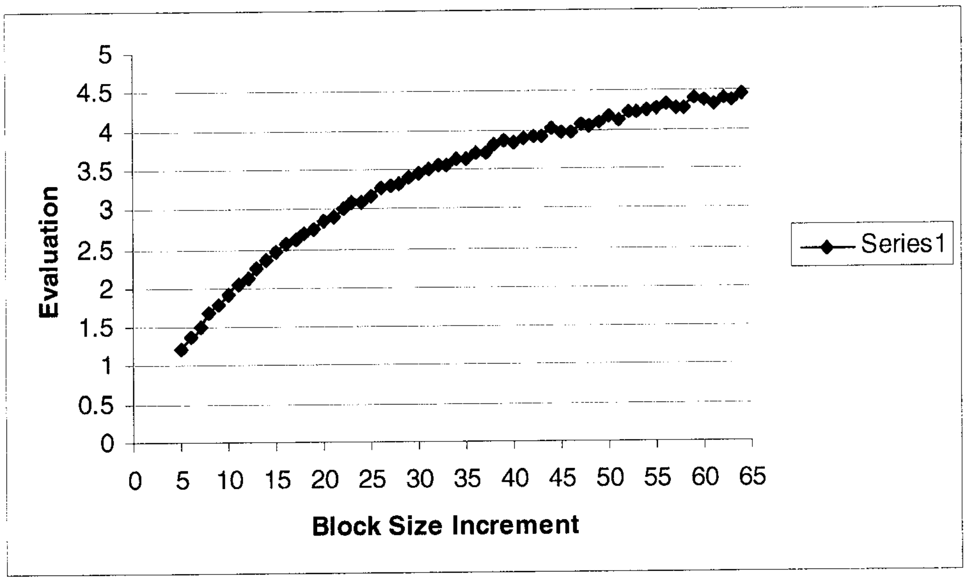


Figure 7-11 Standard Algorithm (10000 Bits – 100 Errors) with basic block size=10

a was calculated and it turned out to be 4.46

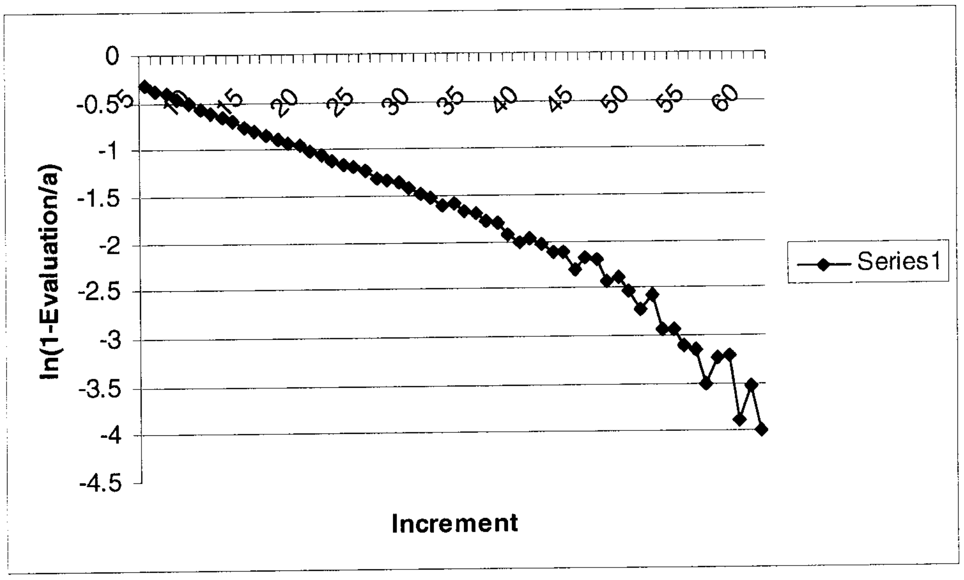


Figure 7-12 Standard Algorithm (10000 Bits – 100 Errors)
Block Size Inc. vs. ln (1-Eval/a)

b was calculated according to equation 7.2 and it turned out to be 0.001203714

Various different experiments were calculated by changing the block size increment, basic block size and the number of discrepancies between the sender's and receiver's bits. The results are combined and are presented at the end of this chapter.

7.3.2 Enhanced Algorithm

The experiments and values obtained on the standard algorithm are repeated on the enhanced algorithm to be able to get the corresponding values and graphs. The results of running the experiment in a 10000-bits key string with 50 errors and a basic block size of 10 are shown in Fig. 7.13

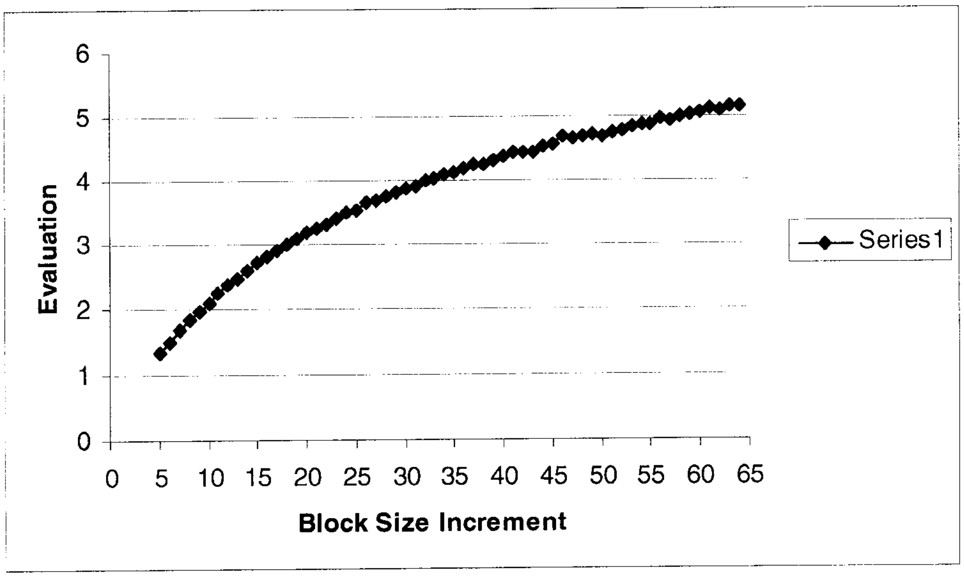


Figure 7-13 Enhanced Algorithm (10000 Bits – 50 Errors) with basic block size=10

a was calculated and it turned out to be 5.17 . Plotting $\ln(1-\frac{\text{Evaluation}}{a})$ versus block size yielded Fig. 7.14

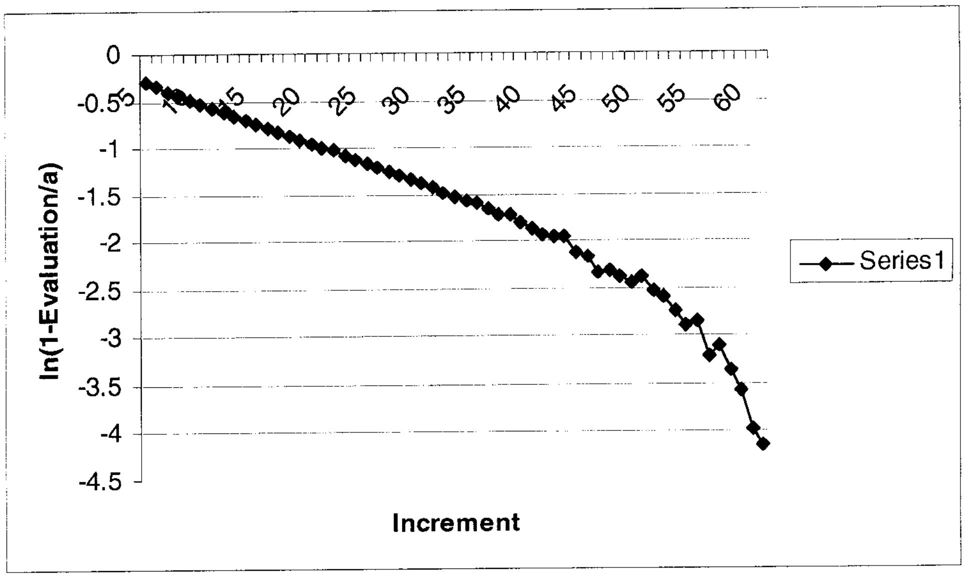


Figure 7-14 Enhanced Algorithm (10000 Bits – 50 Errors)

Block Size Inc. vs. ln (1-Eval/a)

b was calculated according to equation 7.2 and it turned out to 0.001248532

The experiment was re-run again on the enhanced algorithm but with 100 errors between the sender's and the receiver's keys (rather than 50 errors in the first experiment). Figures 7.15 and 7.16 are plotted for this experiment.

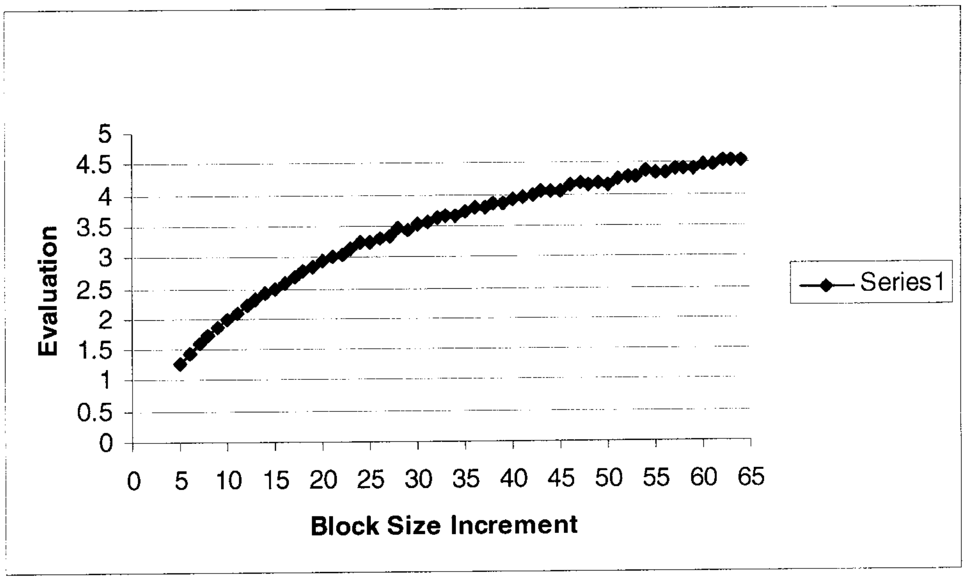


Figure 7-15 Enhanced Algorithm (10000 Bits – 100 Errors) with basic block size=10

a was calculated and it turned out to be 4.55

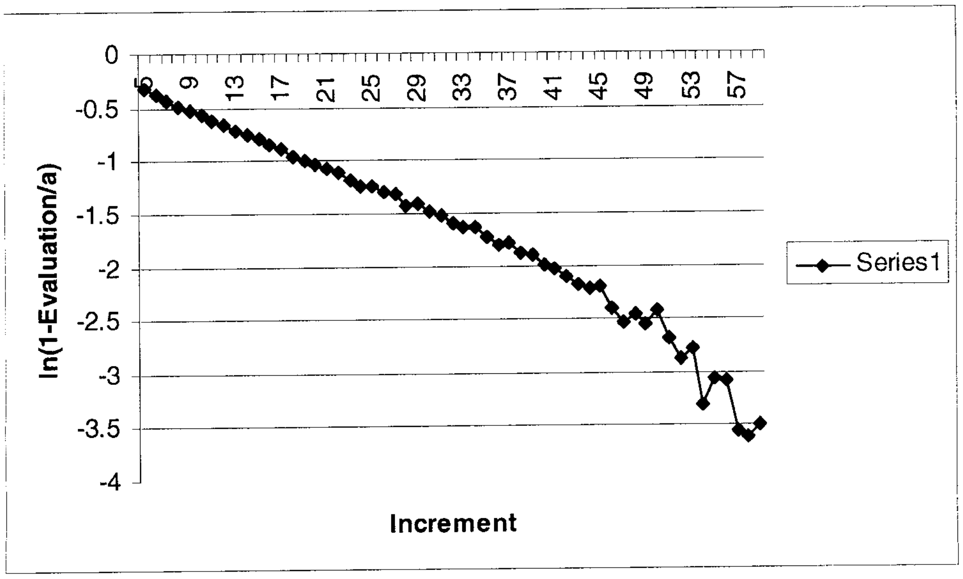


Figure 7-16 Enhanced Algorithm (10000 Bits – 100 Errors)

Block Size Inc. vs. ln (1-Eval/a)

b was calculated according to equation 7.2 and it turned out to 0.001284135

As with the standard algorithm, various different experiments were calculated by changing the block size increment, basic block size and the number of discrepancies between the sender's and receiver's bits and the results are combined and are presented in the next section.

7.4 Standard vs. Enhanced according to the mathematical model

A full set of experiments have been made on the standard and the enhanced algorithms using the mathematical model described in this chapter. The experiments were done using 5000 and 10000 bits key strings with various ranges of basic block size and block size increment values. Those experiments followed the same path and had the same graphs as the ones described in the last section "Applying Mathematical Model on Results"; however, not all the graphs were included in this document as they would be redundant. The graphs and the equations of the preceding chapter were used to create a table containing the values of **a** and **b** for the different experiments done (**a** and **b** are the constants of the equation $Y = a(1 - e^{-bX})$. The table of **a,b** values are presented in Table 7.1

Table 7-1 a,b values

	Standard		**enhanced**	
	a	b	a	b
5000 Bits				
100 Error [Incerement Changes]	3.56	0.05683	3.62	0.05911
200 Error [Incerement Changes]	2.35	0.06352	2.36	0.07058
100 Error [Block Size Changes]	3.15	0.02202	3.62	0.02119
200 Error [Block Size Changes]	1.715	0.03203	2.1	0.02248
10000 Bits				
50 Error [Incerement Changes]	5.06	0.001174	5.171	0.001249
100 Error [Incerement Changes]	4.46	0.001204	4.55	0.001284
150 Error [Incerement Changes]	3.95	0.001278	4.1	0.001095
250 Error [Incerement Changes]	3.145	0.001364	3.28	0.001177
300 Error [Incerement Changes]	2.825	0.001343	2.88	0.001352
350 Error [Incerement Changes]	2.58	0.00132	2.628	0.001389
50 Error [Block Size Changes]	6.6	0.000345	7.54	0.000341
100 Error [Block Size Changes]	4.9	0.000415	5.622	0.000403
150 Error [Block Size Changes]	3.87	0.000431	4.464	0.00042
250 Error [Block Size Changes]	2.61	0.000497	3.13	0.000404
300 Error [Block Size Changes]	2.251	0.000484	2.923	0.000332
350 Error [Block Size Changes]	1.945	0.000533	2.48	0.00036

Many observations could be concluded from table 7-1 above.

The first observation would be the changes in the value of **a** when using the enhanced algorithm. We notice that the enhanced algorithm has higher values for **a** than the corresponding standard algorithm under the same conditions. **a** is the maximum value that the evaluation function reaches. This means that the enhanced algorithm gets better evaluation (longer key in less iterations) than the standard algorithm since it has a higher **a**. This is a quantitative measure that shows the performance of the enhanced algorithm in comparison to the standard algorithm under the same conditions. The difference of the value of **a** is above 0.5 in average.

The second observation that can be noted is that the value of **a** decreases with the increase of the number of errors. The observation that the evaluation decrease with the increase in the number of errors has been made before in several experiments; however, this is the first time to have a value to associate it with. This observation is logical because we expect the algorithm to create a smaller key when the number of discrepancies between the initial keys at the sender and the receiver side is larger. The value that **a** decreases when the errors increase varies according to the amount of increase in the error rate and the present error rate; however, the value decreased doesn't depend on the fact that the algorithm used is standard or enhanced.

The value of **b** is generally better in the enhanced algorithm when the basic block size is kept constant and the block size increment is varied; however, when the block size increment is kept constant and the basic block size is varied the opposite occurs and **b** is better in the standard algorithm. The change in the value of **b** is small as the value of **b** itself is rather small. **b** is a representative of the speed the algorithm tries to reaches its best evaluation value.

Chapter 8

Conclusion

&

Future Work

Chapter 8: Conclusion & Future Work

8.1 Conclusion

This research had the objective of making enhancements in the error correction phase of the BB84 protocol to make it more efficient and thus to decrease the number of bits lost during this phase. This enhancement is made through the use of memory between the rounds of error correction where a round would identify the locations it found errors in to help the next round to be more focused. This would decrease the number of rounds used to reach the final shared string (increase the shared string size). In order to be able to apply the modification, an implementation was made to the standard BB84 protocol as well as the enhanced one that uses memory between rounds. Experiments were done to both the standard and the enhanced algorithms in order to assess the effect of the enhancements that have been introduced (those experiments are detailed in chapter 6). The superiority of the present enhanced algorithm appeared in those experiments and appeared in the comparison figures 6-8, 6-9, 6-10, 6-20, 6-21 and 6-22.

When using 2000 bits as the initial key string between Alice and Bob and having 80 discrepancies between them, the comparison graphs made it clear that at the initial block size of 5, there is no significant difference between the results obtained by the standard and the enhanced algorithms. However, when increasing the basic block size to 10 or 20, the gap between the standard and the enhanced algorithm performances became wider and the superiority of the enhanced algorithm became apparent. Using basic block size of 10, the standard algorithm reached a maximum evaluation value of 2.5; however, the enhanced algorithm gave results at the range of 4-5 which is much better. Using basic block size of 20, the standard algorithm works in the range of 2.5 maximum but the enhanced algorithm has reached even a better performance result at the range of 6-8. When using 5000 bits as the initial key string between Alice and Bob and having 100 discrepancies between them, the comparison graphs showed that at the initial block size of 10, there is some difference between the results obtained by the standard and the enhanced algorithm. However, when increasing the basic block size to 20 or 30, the gap between the standard and the enhanced algorithm performances became wider. Using basic block size of 10, the standard algorithm reached a maximum evaluation value of 4.7 and the enhanced algorithm gave results that reached 5.4. Using basic block size of 20, the standard algorithm works in the range of 4.8 maximum and the enhanced algorithm has reached even a better performance result at the range of 6-7.8. Using basic block size of 30, the standard algorithm stayed in the same range of evaluation values, but the enhanced algorithm was able to reach 8.6

Moreover, the superiority of the present enhanced algorithm has been established through the mathematical model that has been formulated for the system (chapter 7). This is apparent in Table 7-1 which compares the values of **a** and **b** for both the standard and the enhanced error correction phase of the BB84 protocol. The enhanced algorithm had higher values for **a** than the corresponding standard algorithm under the same conditions. Since **a** is the maximum value that the evaluation function reaches, then this means that the enhanced algorithm gets better evaluation (longer key in less iterations) than the standard algorithm since it has a higher **a**. This is a quantitative measure that shows the performance of the enhanced algorithm in comparison to the standard algorithm under the same conditions. The difference of the value of **a** between the standard and the enhanced algorithms is above 0.5 in average.

In conclusion, this research provides an enhanced implementation of the standard error correction phase in the BB84 protocol. The enhancement has been proven superior to the standard one in terms that it generates a larger key in a small number of rounds (thus less amount of time). The superiority of the enhanced algorithm has been established through raw result of experiments and through a mathematical model that has been built for the system.

8.2 Future Work

There is currently a set of projects working on realizing Quantum Cryptography and trying to overcome the problems it is facing. Some of those projects are mentioned in chapter 3.5. The most important of these projects is the Najavo project as it became the first commercial application for Quantum Cryptography. Another important project is the ARC project that aims at creating another commercial application using Quantum Cryptography.

Quantum Cryptography faces a set of problems at the lowest levels (the physics level which is the first phase of the BB84 protocol) which are effect of noise, the distance limitation and the photon polarization problems. All these issues should be solved and settled on for Quantum Cryptography to reach its full potential. The current projects running intend to try to resolve some of such issues such as the ARC project that aims at trying to find a workaround for the distance limitation and the photon polarization issues. However, more research needs to be done in those areas to be able to find better solutions for these problems.

Moreover, an enhancement needs to be made to the privacy amplification phase of the BB84 protocol. The privacy amplification phase is concerned with distilling highly secret shared information from a larger amount of shared information that is partially secret. Using the BB84 protocol, Eve might have been able to spy on some of the information sent on the channel and thus might be able to know the value of some part of the shared key between Alice and Bob. That is why the privacy amplification phase is needed.

This stage causes the loss of a large number of bits of the key between the sender and the receiver. This was apparent in the experiments done by Bennet and Brassard where 625 bits were removed during this phase in one of the experiments and 902 were removed in another experiment (see section 4.4). Thus, more research should be done in the privacy amplification phase to enhance it so that it would loose a small number of bits. Such an enhancement in the privacy amplification phase would result in a large improvement in the performance of Quantum Key Distribution and would result in much longer keys to be used in encrypting messages.

The error correction phase can be modified too to become even more efficient than the way presented here. This would mean the re-thinking of the whole phase and how it is to be implemented. Using other measures to check the similarity of the bits at the sender's and the receiver's sides (rather than parity check) and devising a different algorithm for implementing the error correction phase might prove to bear better results.

References

References

BENNET, Charles., BRASSARD, Gilles., “Quantum Cryptography: Public Key Distribution and Coin Tossing”, Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing, Bangalore, India (IEEE, New York, 1984) p.175.

BENNET, C.H., BRASSARD, G., and ROBERT, G.M., “Privacy Amplification by public discussion”, SIAM Journal of Computing, Vol. 17 no.2, (April 1988.), p.210-229,

BENNET, Charles., BESSETTE, Francois., BRASSARD, Gilles. , SALVAIL, Louis., and SMOLIN , John., “Experimental Quantum Cryptography”, Journal of Cryptography, Vol.5 , (1992), p. 3-28.

BENNET, Charles H., “Quantum Cryptography using any two Nonorthogonal States”. Physical Review Letters, Vol 68 no.21, (25 May 1992), p.3121-3124.

BENNET, Charles., BRASSARD, Gilles., CREPEAU, Claude, MAURER, Ueli M., “Generalized Privacy Amplification”, IEEE Transaction on Information Theory, Vol. 41, no. 6, (1995) pp. 1915-1923.

BOYER, Michel., BEHAM, Eli., BRASSARD, Gilles., VAN DE GRAAF, Jeoren., and MOR, Tal., “Security of Quantum Key Distribution Against All Collective Attacks”, Tech. Rep. 9801022, LANL Quant-ph archives(December 5, 2001).

BRASSARD, G., and CREPEAU, C., JOSZA, R., LANGLOIS, Daniel, A quantum bit commitment scheme provably unbreakable by both parties”, Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science, (November 1993) , p.361-371.

BRASSARD, G., and CREPEAU, C. “25 years of quantum cryptography”, SIGACT News, Vol. 27, (3 September 1996), p. 13-24.

BRASSARD, Gilles., CREPEAU, Claude., MAYERS, Dominic., and SALVAIL, Louis, “The Security of Quantum Bit Commitment Schemes”, Proceedings of Randomized Algorithms, Satellite Workshop of 23rd International Symposium on Mathematical Foundations of Computer Science, Brno, (August1998), p. 13-15.

BRASSARD, Gilles. ,”A Bibliography of Quantum Cryptography”. http://www.cs.mcgill.ca/~crepeau/CRYPTO/Biblio-QC.html , 1994

BRYLEVSKI, Alexei. “Quantum Key Distribution: real Time Compensation of Interferometer Phase Drift”, http://www.vad1.com/qcr/alexey/, NTNU University, February 2002.

BUTTLER, W. T., HUGHES, R. T., KWAIT, P.G., LAMOREAUX, S. K. , LUTHER, G. G., MORGAN G. L., NORDHOLT J. E., PETERSON C. G., and SIMMONS C., "Practical free-space quantum key distribution over 1 km", Physics Review Letters, Vol. 81, (1998) p. 3283-3286.

BUTTLER, W. T., HUGHES, R. T. , LAMOREAUX, S. K. , MORGAN G. L., NORDHOLT J. E., PETERSON C. G., "Daylight quantum key distribution over 1.6 km", Physics Review Letters, Vol. 84 (2000) p. 5652-5655.

CASTELLETTO, S. , DEGIOVANNI, L.P., RASTELLO, M. L., "Effects of Experimental Limits in Quantum Cryptography Systems based on polarization entangled photons", http://www.physics.umd.edu/rgroups/ep/yskim/boston/castel.pdf Istituto Elettrotecino Nazionale Torino ,Italy (2 January 2002).

DIFFIE, W., and HELLMAN, M.E., "New Directions in Cryptography", IEEE Transactions on Information Theory IT-22, (1976) p.644-654.

EKERT, A. k., "Qunatum Cryptography Based On Belles's Theorem", Physical Review Letters, Vol.67 no.6, (5 August 1991), p.661-663.

FORD, James, "Quantum Cryptography Tutorial", http://www.cs.dartmouth.edu/~jford/crypto.html , 1996.

FRANSON, J.D., JACCOBS, B. C., "Operational System for Quantum Cryptography", Elec. Letters, vol. 31, 1995.

GISIN, Nicolas., RIDORDY, Gregoire., TITTLE, Wolfgang., ZBINDEN, Hugo, "Quantum Cryptography", Reviews of Modern Physics, Vol. 74, (January 2002), p. 145-195.

GOLDENBERG, Lior , VAIDMAN, Lev, "Quantum Cryptography Based on Orthogonal States". Physical Letters Review, Vol. 75 NO 7, (14 August 1995),p. 1239-1243.

GORMAN, P. M., TAPSTER, P.R., and RARITY, J. G., "Secure free-space key exchange to 1.9 km and Beyond", Journal of Modern Optics, Vol. 48, (2001), p.1887-1901,.

GROVER, L. K., "A framework for fast quantum mechanical algorithms", Proceedings of the 30th annual ACM Symposium on the Theory of computing, (1998) p. 53-62.

HUGHES, R.J, LUTHER, C.G., MORGAN, G.L., PETERSON, C.G., SIMMONS, C., "Quantum Cryptography over underground fibers", Proceedings of Crypto 1996, Lecture Notes in Computer Science, Vol. 1109, (1996) p.329.

KOLLMITZER Ch. , MONYK, CH., PEEV M., SUDA M. "An Advance towards Practical Quantum Cryptography", Austrian Research Centers (2002), http://www.arcs.ac.at/quanteninfo

MagiqTech Navajo White Paper, "Perfectly Secure Key Management System Using Quantum Key Distribution: Code-Name: Navajo". http://magiqtech.com/, 2003

MagiqTech. Backgrounder, "MagiqTech Corporate Backgrounder 2002". http://magiqtech.com/, 2002

MAYERS, Dominic, "Unconditionally secure quantum bit commitment is impossible", Physical Review Letters, Vol. 78, No. 17, (April 1997), p. 3414-3417.

MAYERS, Dominic, "Unconditional Security in Quantum Cryptography", Journal of the ACM, Vol 48, No. 3, (May 2001).

Molin , Richard A. An Introduction to Cryptography, Chapman & Hall/CRC 2001.

MULLER, A., ZBINDEN, H., and GISIN, N., "Quantum Cryptography over 23 km in installed under-lake telecom fibre", Europhysics Letters, Vol. 33, 1996.

"Quantum Cryptanalysis". http://www.milketoast.com/school/cryptanalysis.htm, (October 25, 2002).

RARITY, J. G., OWENS, P. C. M., and Tapster P.R., "Quantum random-number generation and key-sharing", Journal of Mod. Opt. Vol.41 (1994), p. 2435-2444

RIEFFEL ,Eleanor and POLAK, Wolfgang., "An Introduction to Quantum Computing for Non-Physicists", ACM Computing Surveys, Vol 32 No. 3, (September 2000), P.300-335.

SCHNEIER, Bruce, Applied Cryptography: Protocols, Algorithms and Source Code in C Second Edition, Wiely 1996.

SCHULMAN, L. J. and VAZIRANI, U., "Scalable NMR Quantum Computation", Los Alamos Physics Archive, 1998.

SHOR, P.W., "Algorithms for quantum computation: Discrete log and factoring", Proceedings of the 35th annual symposium on foundations of Computer Science, (Nov. 1994) p.124-134.

SHOR, P.W., "Polynomial-time algorithms for prime factorization and discrete logarithms in a quantum computer", Society for Industrial and Applied Mathematics Journal of Computing, Vol. 26 no. 5, (1997),p. 1484-1509.

TOWNSEND, P.D., RARITY, J. G. and TAPSTER, P. R., "Enhanced Single photon fringe visibility in a 10 km-long prototype quantum cryptography channel", Electronics Letters, vol. 29, (8 July 1993), p.1291-1293.

TOWNSEND, Paul D., "Simultaneous Quantum cryptographic key distribution and conventional data transmission over installed fibre using WDM", Electrical Letters, Vol. 33, (1997).

WITTENBERG, David K. "Reducing the Randomness Requirements for Quantum Money". Technical Report, CS Department, Brandeis University, (23 January 1995).

XU, ZHOU, "An introduction to Quantum Key Distribution", www.comp.nus.edu.sg/~xuzhou/reports/quantum-cryptography-survey-xuzhou-11-2002.pdf, (October 2002)

ZBINDEN, H. , GISIN, N. , HUTTENR, B. , MULLER, A. , TITTLE, W. "Practical Aspects of Quantum Cryptographic Key Distribution". Journal Of Cryptography Vol. 13 (2000) p. 207-220.

Appendix A

Tables and Graphs

Appendix A: Tables and Graphs

This appendix contains a set of tables and graphs that were obtained by running both the standard and the enhanced algorithms on different key values with different parameters. These tables and graphs are referred to in chapter 6 "Test Results and Assessment" and they are the basis for the comparison graphs 6-4, 6-5, 6-6, 6-10, 6-11and 6-12.

Table A-1 Standard Algorithm (2000 Bits – 80 Errors) with basic block size =10

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
10	5	13	807	1193	333	1.478314746	73	9.733735784	82.733736	4.136686789
10	10	15	729	1271	345	1.743484225	73	9.081574035	82.081574	4.104078702
10	15	14	660	1340	347	2.03030303	73	10.55065072	83.550651	4.177532536
10	20	15	650	1350	370	2.076923077	74	9.061458528	83.061459	4.153072926
10	25	17	646	1354	381	2.095975232	73	7.706736935	80.706737	4.035336847
10	30	14	609	1391	376	2.28407225	74	9.741041582	83.741042	4.187052079
10	35	15	607	1393	383	2.294892916	74	8.574525024	82.574525	4.128726251
10	40	16	599	1401	381	2.338898164	70	7.407217692	77.407218	3.870360885
10	45	14	574	1426	374	2.484320557	73	10.48840515	83.488405	4.174420257
10	50	17	606	1394	399	2.300330033	74	9.309116687	83.309117	4.165455834
10	55	16	586	1414	387	2.412969283	72	8.633354221	80.633354	4.031667711
10	60	20	609	1391	398	2.28407225	67	6.71275063	73.712751	3.685637531
10	65	16	599	1401	408	2.338898164	73	8.413076862	81.413077	4.070653843
10	70	20	613	1387	408	2.262642741	71	7.301390919	78.301391	3.915069546
10	75	21	614	1386	410	2.25732899	73	9.699631715	82.699632	4.134981586
10	80	19	615	1385	420	2.25203252	73	8.580441591	81.580442	4.07902208
10	85	22	640	1360	435	2.125	73	7.806972117	80.806972	4.040348606
10	90	26	612	1388	399	2.267973856	68	7.322305937	75.322306	3.766115297
10	95	27	628	1372	419	2.184713376	66	5.337892999	71.337893	3.56689465
10	100	25	628	1372	415	2.184713376	69	7.010762661	76.010763	3.800538133

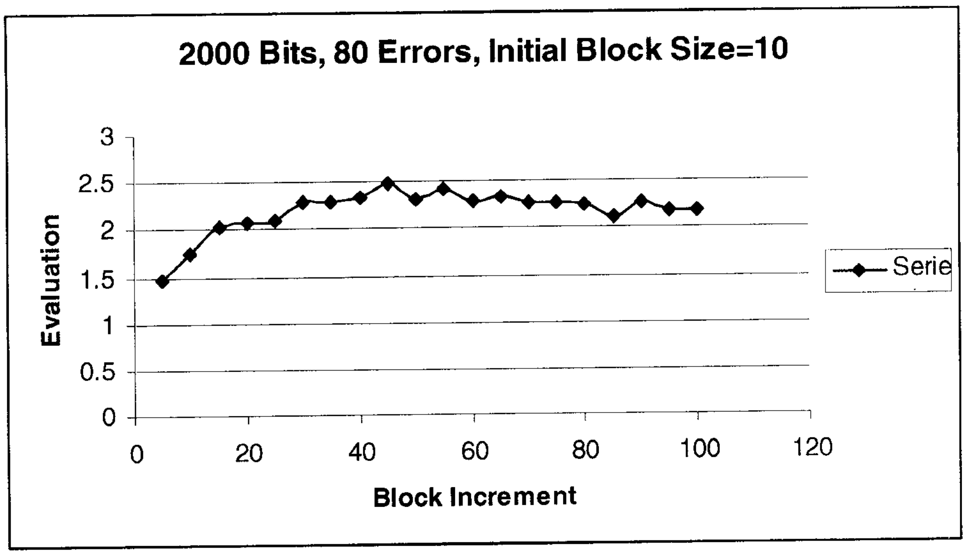


Figure A-1 Standard Algorithm (2000 Bits – 80 Errors) with basic block size =10

Table A-2 Standard Algorithm (2000 Bits – 80 Errors) with basic block size =20

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
20	10	15	729	1271	345	1.743484	73	8.380431	81.38043	4.069022
20	30	17	630	1370	383	2.174603	71	6.915772	77.91577	3.895789
20	50	16	589	1411	386	2.395586	71	8.000511	79.00051	3.950026
20	70	19	598	1402	398	2.344482	71	8.498276	79.49828	3.974914
20	90	20	601	1399	408	2.327787	71	7.609736	78.60974	3.930487
20	110	26	623	1377	409	2.210273	69	7.375224	76.37522	3.818761
20	130	32	634	1366	423	2.154574	64	4.44395	68.44395	3.422197
20	150	22	605	1395	417	2.305785	71	7.939276	78.93928	3.946964
20	170	24	581	1419	400	2.442341	67	8.171251	75.17125	3.758563
20	190	27	617	1383	431	2.241491	69	6.878253	75.87825	3.793913
20	210	20	595	1405	415	2.361345	72	9.258178	81.25818	4.062909
20	230	24	630	1370	446	2.174603	71	7.573861	78.57386	3.928693
20	250	23	585	1415	410	2.418803	67	7.469407	74.46941	3.72347
20	270	26	564	1436	386	2.546099	66	7.820693	73.82069	3.691035
20	290	26	602	1398	424	2.322259	68	7.356466	75.35647	3.767823
20	310	28	576	1424	392	2.472222	61	5.694139	66.69414	3.334707
20	330	30	641	1359	452	2.120125	70	6.831593	76.83159	3.84158
20	350	34	672	1328	487	1.97619	69	5.435459	74.43546	3.721773
20	370	34	642	1358	462	2.115265	70	6.801919	76.80192	3.840096
20	390	27	542	1458	381	2.690037	60	6.255961	66.25596	3.312798

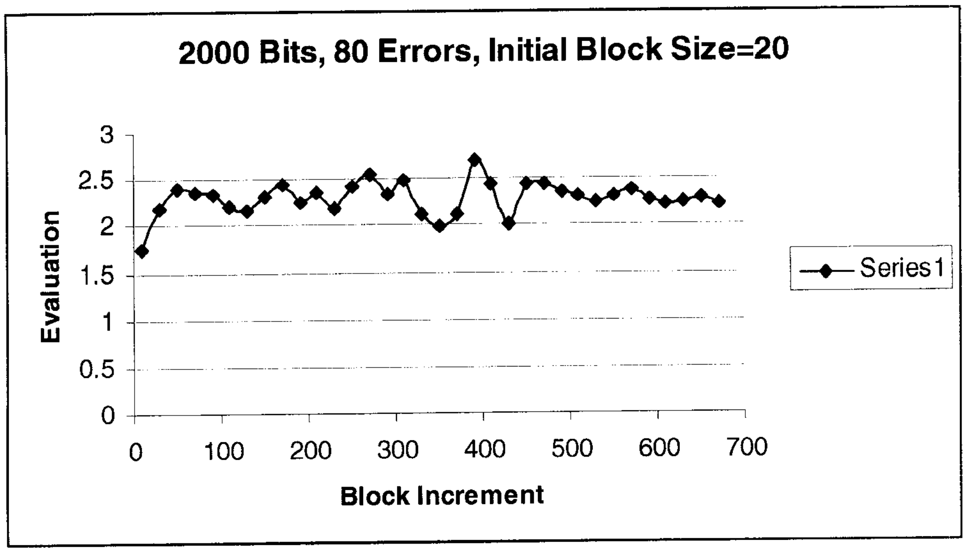
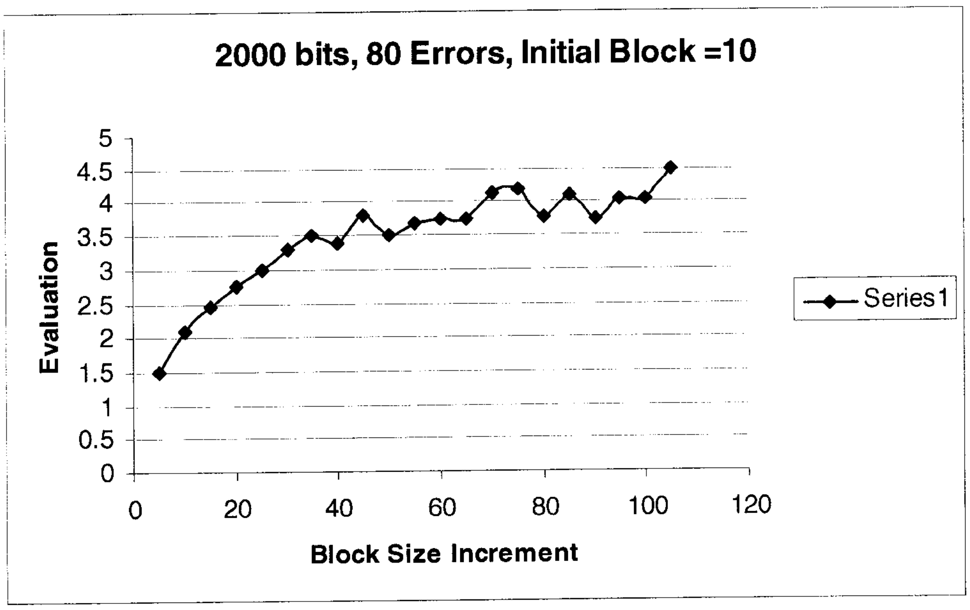


Figure A-2 Standard Algorithm (2000 Bits – 80 Errors) with basic block size =20

Table A-3 Enhanced Algorithm (2000 Bits – 80 Errors) with basic block size =10

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
10	5	12	886	1320	207	1.489841986	60	6.703724291	66.703724	3.335186215
10	10	13	707	1481	189	2.09476662	56	6.256809339	62.256809	3.112840467
10	15	13	634	1557	192	2.455835962	56	6.256809339	62.256809	3.112840467
10	20	14	584	1604	189	2.746575342	54	6.033351862	60.033352	3.001667593
10	25	15	547	1635	183	2.989031079	54	6.033351862	60.033352	3.001667593
10	30	14	504	1659	164	3.291666667	48	5.362979433	53.362979	2.668148972
10	35	15	479	1676	156	3.498956159	46	5.139521957	51.139522	2.556976098
10	40	16	498	1689	188	3.391566265	56	6.256809339	62.256809	3.112840467
10	45	16	446	1701	148	3.813901345	46	5.139521957	51.139522	2.556976098
10	50	16	489	1707	197	3.490797546	58	6.480266815	64.480267	3.224013341
10	55	16	467	1717	185	3.676659529	56	6.256809339	62.256809	3.112840467
10	60	17	460	1721	182	3.741304348	52	5.809894386	57.809894	2.890494719
10	65	17	460	1728	189	3.756521739	56	6.256809339	62.256809	3.112840467
10	70	17	418	1732	151	4.14354067	44	4.91606448	48.916064	2.445803224
10	75	18	413	1733	147	4.196125908	44	4.91606448	48.916064	2.445803224
10	80	19	459	1737	197	3.784313725	58	6.480266815	64.480267	3.224013341
10	85	20	424	1743	168	4.110849057	50	5.586436909	55.586437	2.779321845
10	90	20	467	1742	210	3.730192719	62	6.927181768	68.927182	3.446359088
10	95	21	432	1744	177	4.037037037	52	5.809894386	57.809894	2.890494719
10	100	22	432	1747	180	4.043981481	54	6.033351862	60.033352	3.001667593



FigureA-3 Enhanced Algorithm (2000 Bits – 80 Errors) with basic block size =10

Table A-4 Enhanced Algorithm (2000 Bits – 80 Errors) with basic block size =20

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
20	10	14	559	1630	190	2.915921288	44	2.340179042	46.34017904	2.317008952
20	30	16	417	1768	186	4.239808153	42	2.233807267	44.23380727	2.211690363
20	50	15	363	1810	174	4.986225895	40	2.127435492	42.12743549	2.106371775
20	70	16	329	1832	162	5.568389058	38	2.021063718	40.02106372	2.001053186
20	90	17	293	1842	136	6.28668942	32	1.701948394	33.70194839	1.68509742
20	110	18	302	1853	156	6.135761589	36	1.914691943	37.91469194	1.895734597
20	130	20	299	1857	157	6.210702341	36	1.914691943	37.91469194	1.895734597
20	150	19	342	1862	205	5.444444444	46	2.446550816	48.44655082	2.422327541
20	170	20	279	1862	142	6.673835125	32	1.701948394	33.70194839	1.68509742
20	190	21	299	1865	165	6.237458194	38	2.021063718	40.02106372	2.001053186
20	210	21	302	1869	172	6.188741722	38	2.021063718	40.02106372	2.001053186
20	230	22	302	1866	169	6.178807947	40	2.127435492	42.12743549	2.106371775
20	250	23	282	1866	149	6.617021277	34	1.808320169	35.80832017	1.790416008
20	270	22	308	1869	178	6.068181818	40	2.127435492	42.12743549	2.106371775
20	290	23	315	1869	185	5.933333333	42	2.233807267	44.23380727	2.211690363
20	310	24	296	1876	173	6.337837838	40	2.127435492	42.12743549	2.106371775
20	330	24	316	1876	193	5.936708861	44	2.340179042	46.34017904	2.317008952
20	350	25	312	1876	189	6.012820513	42	2.233807267	44.23380727	2.211690363
20	370	36	274	1879	154	6.857664234	36	1.914691943	37.91469194	1.895734597
20	390	26	306	1879	186	6.140522876	42	2.233807267	44.23380727	2.211690363

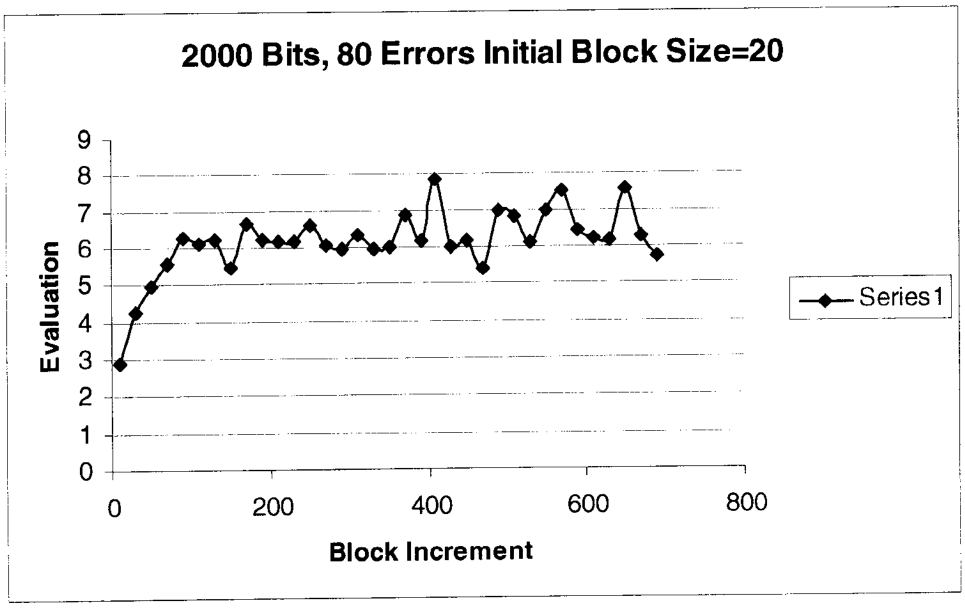


Figure A-4 Enhanced Algorithm (2000 Bits – 80 Errors) with basic block size =20

Table A-5 Standard Algorithm (5000 Bits – 100 Errors) with basic block size =20

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
20	5	14	1741	3259	444	1.871912694	98	9.518770923	107.5187709	2.150375418
20	10	13	1415	3585	459	2.533568905	97	9.36674707	106.3667471	2.127334941
20	20	14	1191	3809	470	3.198152813	97	8.335477961	105.335478	2.106709559
20	30	15	1079	3921	469	3.633920297	96	8.673063966	104.673064	2.093461279
20	40	15	1029	3971	494	3.859086492	99	9.653986366	108.6539864	2.173079727
20	50	15	1016	3984	527	3.921259843	99	8.10873843	107.1087384	2.142174769
20	60	14	946	4054	495	4.285412262	98	9.077419076	107.0774191	2.141548382
20	70	14	919	4081	491	4.440696409	94	8.262601423	102.2626014	2.045252028
20	80	14	919	4081	510	4.440696409	97	9.022432057	106.0224321	2.120448641
20	90	16	920	4080	519	4.434782609	97	8.46902426	105.4690243	2.109380485
20	100	17	948	4052	557	4.274261603	97	7.412793825	104.4127938	2.088255877
20	110	14	883	4117	510	4.662514156	98	10.17066784	108.1706678	2.163413357
20	120	16	914	4086	543	4.470459519	96	7.965576562	103.9655766	2.079311531
20	130	15	903	4097	542	4.53709856	99	9.291842916	108.2918429	2.165836858
20	140	21	973	4027	593	4.138746146	97	6.715152001	103.715152	2.07430304
20	150	14	859	4141	513	4.820721769	99	10.91062097	109.910621	2.198212419

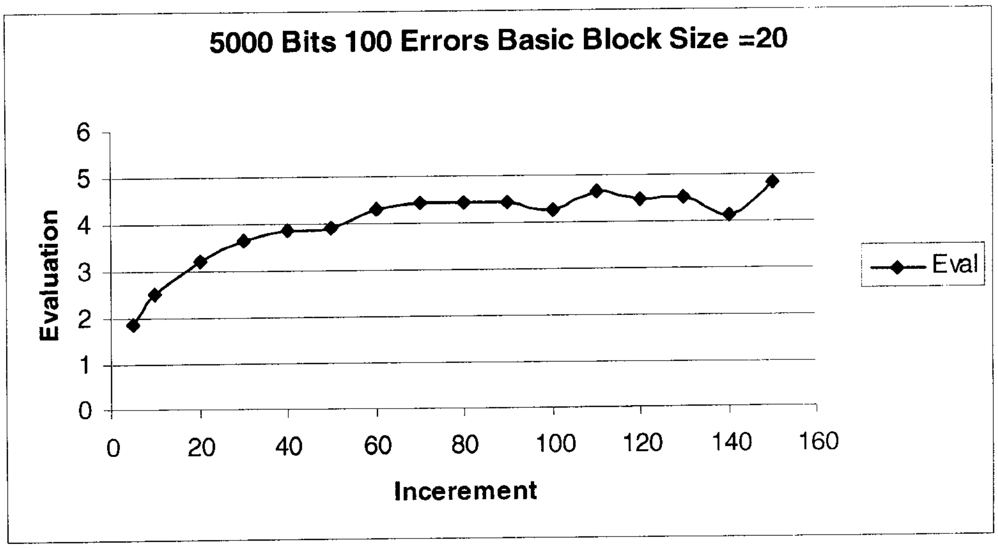


Figure A-5 Standard Algorithm (5000 Bits – 100 Errors) with basic block size =20

Table A-6 Standard Algorithm (5000 Bits – 100 Errors) with basic block size =30

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
30	5	13	1697	3303	442	1.946375958	97	9.614410469	106.6144105	2.132288209
30	10	13	1413	3587	458	2.538570418	99	10.36812676	109.3681268	2.187362535
30	20	14	1163	3837	444	3.299226139	96	10.11758137	106.1175814	2.122351627
30	30	14	1077	3923	476	3.642525534	96	9.005653213	105.0056532	2.100113064
30	40	15	1023	3977	487	3.887585533	98	8.582259126	106.5822591	2.131645183
30	50	14	972	4028	488	4.144032922	97	9.640746964	106.640747	2.132814939
30	60	14	937	4063	486	4.336179296	97	9.370717547	106.3707175	2.127414351
30	70	14	935	4065	507	4.347593583	96	9.15647147	105.1564715	2.103129429
30	80	16	947	4053	530	4.279831045	96	7.97151678	103.9715168	2.079430336
30	90	14	919	4081	527	4.440696409	97	8.828939944	105.8289399	2.116578799
30	100	17	945	4055	551	4.291005291	96	7.592045159	103.5920452	2.071840903
30	110	14	897	4103	524	4.574136009	99	9.603558407	108.6035584	2.172071168
30	120	19	975	4025	591	4.128205128	96	6.873449358	102.8734494	2.057468987
30	130	16	898	4102	533	4.567928731	98	9.323615305	107.3236153	2.146472306
30	140	15	874	4126	519	4.720823799	97	10.16306586	107.1630659	2.143261317
30	150	16	900	4100	545	4.555555556	95	7.537683934	102.5376839	2.050753679

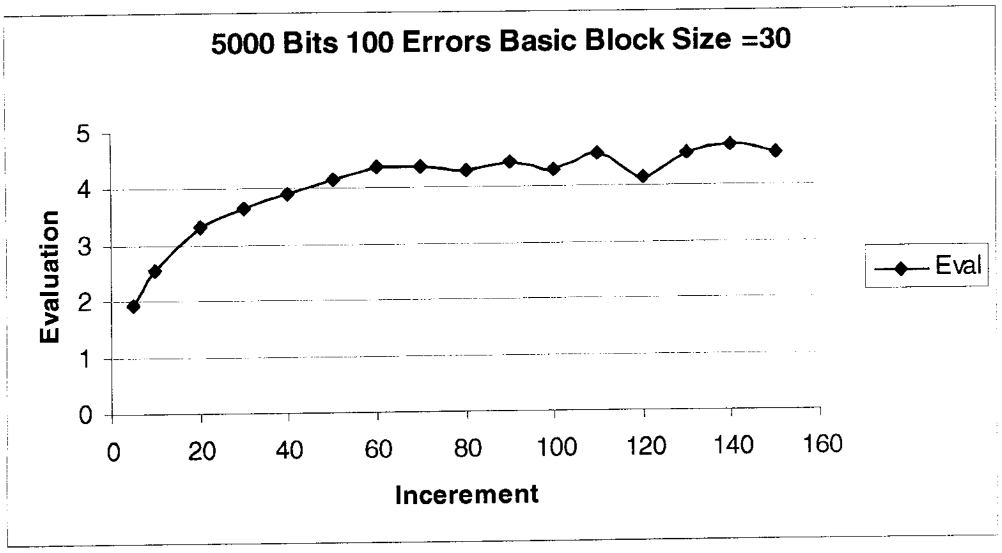


Figure A-6 Standard Algorithm (5000 Bits – 100 Errors) with basic block size =30

Table A-7 Standard Algorithm (5000 Bits – 200 Errors) with basic block size =20

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
20	5	14	2050	2950	838	1.43902439	183	23.45964388	206.4596439	4.129192878
20	10	15	1791	3209	844	1.79173646	181	24.16106933	205.1610693	4.103221387
20	20	16	1646	3354	939	2.037667072	186	21.54171972	207.5417197	4.150834394
20	30	17	1594	3406	995	2.136762861	183	17.60345792	200.6034579	4.012069158
20	40	20	1585	3415	1039	2.154574132	185	18.04895818	203.0489582	4.060979164
20	50	21	1575	3425	1076	2.174603175	183	17.0333524	200.0333524	4.000667048
20	60	21	1581	3419	1114	2.162555345	187	18.03984198	205.039842	4.10079684
20	70	27	1600	3400	1131	2.125	185	18.70815296	203.708153	4.074163059
20	80	31	1600	3400	1134	2.125	186	19.78814384	205.7881438	4.115762877
20	90	33	1606	3394	1143	2.113325031	181	16.68515356	197.6851536	3.953703071
20	100	35	1626	3374	1167	2.07503075	185	18.47241779	203.4724178	4.069448356
20	110	39	1659	3341	1193	2.013863773	186	18.2965483	204.2965483	4.085930966
20	120	41	1671	3329	1205	1.992220227	185	17.94180691	202.9418069	4.058836138
20	130	53	1774	3226	1263	1.81848929	184	15.72864381	199.7286438	3.994572876
20	140	38	1637	3363	1193	2.054367746	177	14.84228982	191.8422898	3.836845796
20	150	43	1708	3292	1247	1.927400468	187	17.75423239	204.7542324	4.095084648

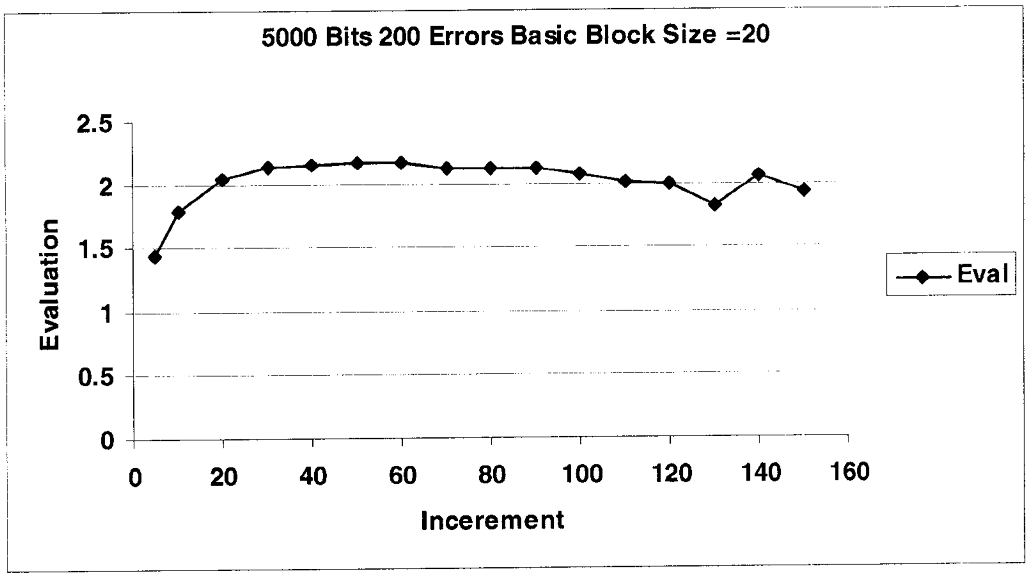


Figure A-7 Standard Algorithm (5000 Bits – 200 Errors) with basic block size =20

Table A-8 Standard Algorithm (5000 Bits – 200 Errors) with basic block size =30

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
30	5	15	2092	2908	847	1.390057361	185	23.03435868	208.0343587	4.160687174
30	10	15	1839	3161	895	1.718868951	187	22.24537626	209.2453763	4.184907525
30	20	16	1648	3352	940	2.033980583	182	19.7002303	201.7002303	4.034004606
30	30	17	1621	3379	1025	2.084515731	192	20.09561096	212.095611	4.241912219
30	40	18	1576	3424	1043	2.172588832	185	18.2876866	203.2876866	4.065753732
30	50	17	1492	3508	1010	2.351206434	181	19.60912073	200.6091207	4.012182415
30	60	21	1533	3467	1066	2.261578604	180	17.17458673	197.1745867	3.943491735
30	70	28	1582	3418	1109	2.160556258	184	19.2221326	203.2221326	4.064442652
30	80	31	1588	3412	1121	2.14861461	184	18.6954961	202.6954961	4.053909922
30	90	38	1702	3298	1219	1.937720329	184	15.17496046	199.1749605	3.983499209
30	100	32	1650	3350	1201	2.03030303	182	15.42664117	197.4266412	3.948532823
30	110	36	1663	3337	1209	2.006614552	183	16.26037754	199.2603775	3.985207551
30	120	41	1667	3333	1199	1.99940012	181	16.06965921	197.0696592	3.941393184
30	130	37	1604	3396	1160	2.117206983	182	18.1301603	200.1301603	4.002603206
30	140	50	1733	3267	1255	1.885170225	178	13.85887371	191.8588737	3.837177474
30	150	45	1750	3250	1281	1.857142857	183	14.10868454	197.1086845	3.942173691

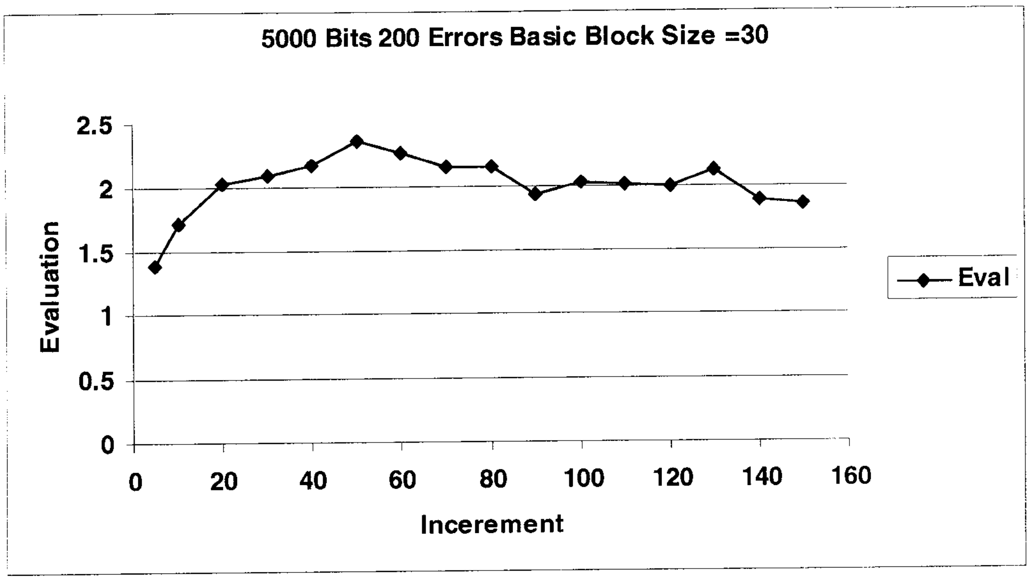


Figure A-8 Standard Algorithm (5000 Bits – 200 Errors) with basic block size =30

Table A-9 Enhanced Algorithm (5000 Bits – 100 Errors) with basic block size =20

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
20	5	12	1374	3624	219	2.637554585	49	4.01701311	53.01701311	1.060340262
20	10	13	1116	3884	219	3.480286738	49	4.120826437	53.12082644	1.062416529
20	20	14	894	4100	224	4.586129754	50	4.481720927	54.48172093	1.089634419
20	30	14	789	4209	227	5.33460076	50	4.71913172	54.71913172	1.094382634
20	40	14	736	4260	233	5.788043478	48	3.406248049	51.40624805	1.028124961
20	50	15	696	4304	237	6.183908046	49	3.741346636	52.74134664	1.054826933
20	60	14	688	4311	259	6.265988372	49	3.160658084	52.16065808	1.043213162
20	70	14	640	4357	233	6.8078125	50	4.440783997	54.440784	1.08881568
20	80	14	629	4370	237	6.947535771	49	3.853241259	52.85324126	1.057064825
20	90	16	635	4361	257	6.867716535	50	3.883764771	53.88376477	1.077675295
20	100	17	598	4401	233	7.359531773	50	4.675152074	54.67515207	1.093503041
20	110	14	604	4391	248	7.26986755	50	3.832335716	53.83233572	1.076646714
20	120	16	596	4400	248	7.382550336	48	3.428767821	51.42876782	1.028575356
20	130	15	597	4401	255	7.371859296	49	3.739402711	52.73940271	1.054788054
20	140	15	591	4406	252	7.455160745	50	4.132449574	54.13244957	1.082648991
20	150	15	573	4427	242	7.72600349	50	4.416196202	54.4161962	1.088323924

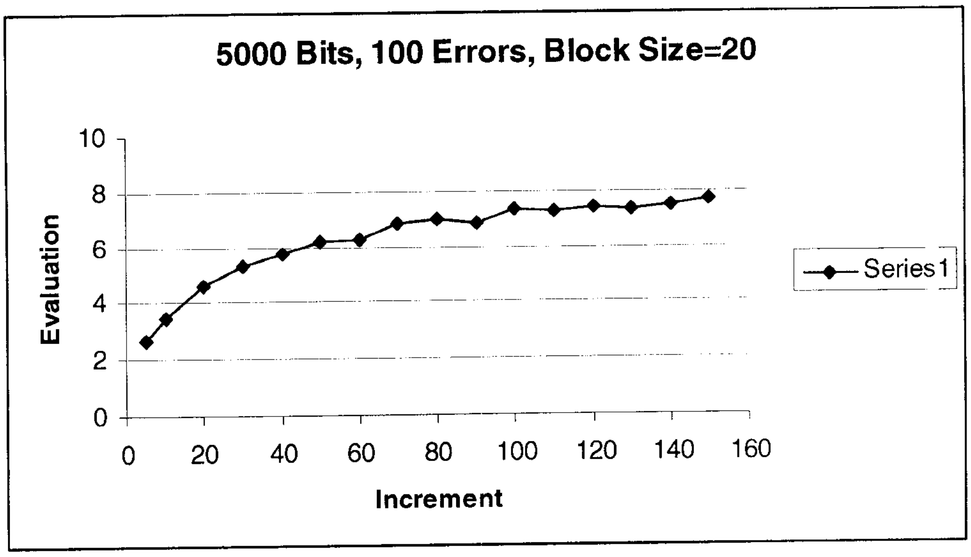


Figure A-9 Enhanced Algorithm (5000 Bits – 100 Errors) with basic block size =20

Table A-10 Enhanced Algorithm (5000 Bits – 100 Errors) with basic block size =30

Basic Block Size	Block Size Increment	Number of Passes	Total Operations	Key Length	Bits Removed	Evaluation	Errors Found	Interpolated Errors	Total Errors	Error Rate
30	5	12	1143	3853	239	3.370953631	48	2.950517165	50.95051716	1.019010343
30	10	12	970	4027	253	4.151546392	50	3.672927155	53.67292716	1.073458543
30	20	13	792	4207	250	5.311868687	48	3.120257659	51.12025766	1.022405153
30	30	14	713	4280	259	6.002805049	50	3.238241219	53.23824122	1.064764824
30	40	14	658	4341	257	6.597264438	49	3.022582621	52.02258262	1.040451652
30	50	15	642	4354	277	6.781931464	50	2.820991322	52.82099132	1.056419826
30	60	15	596	4401	256	7.384228188	49	3.320552352	52.32055235	1.046411047
30	70	14	609	4387	291	7.203612479	50	2.492505053	52.49250505	1.049850101
30	80	15	572	4427	270	7.73951049	50	3.099954493	53.09995449	1.06199909
30	90	14	545	4453	254	8.170642202	47	2.8940203	49.8940203	0.997880406
30	100	15	571	4428	292	7.754816112	50	2.653909994	52.65390999	1.0530782
30	110	16	543	4455	271	8.20441989	50	3.535668704	53.5356687	1.070713374
30	120	17	531	4465	270	8.4086629	50	3.407574898	53.4075749	1.068151498
30	130	16	541	4454	286	8.232902033	49	2.698674298	51.6986743	1.033973486
30	140	17	525	4469	275	8.512380952	49	2.853037033	51.85303703	1.037060741
30	150	18	517	4483	271	8.671179884	48	2.911949898	50.9119499	1.018238998

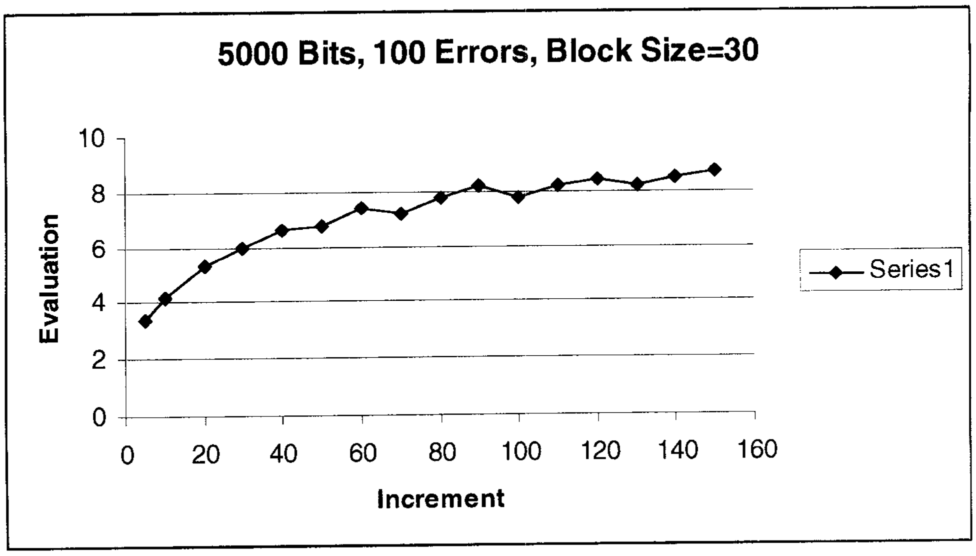


Figure A-10 Enhanced Algorithm (5000 Bits – 100 Errors) with basic block size =30