CRYPTOSCANF: CRYPTOSYSTEM USING CFG

[1]Aishwarya R Parab, [2]Celroy A Ratos, [3]Fiola Carvalho, [4]R. Nageshwar,
[5]Shambhavi S Paradkar
Department of computer engineering, Don Bosco College of engineering, Goa
Email:[1]aishwaryaparab94@gmail.com , [2]celroyratos@gmail.com , [3]fiolacarvalho@gamil.com,
[4]r.nagesh21@gmail.com, [5]paradkar34@gmail.com

Abstract— There is no theoretical study which proves that, given a set of strings from a language how difficult it is to generate another string which belongs to the same language. The fascinating property of CFG is that it is hard to identify the grammar given only the strings generated by it, however it is easy to generate and validate strings from a given grammar.
Due to this property of CFG the idea is to develop a CFG based cryptosystem to provide security for text files. This cryptosystem is a symmetric key encryption technique which consists of various levels of encryption and decryption at sender and receiver side.

***Index Terms*— grammar, cryptosystem, symmetric, CFG, encryption, decryption.**

I. INTRODUCTION

Data security is a challenging issue of data communications today that touches many areas including secure communication channel, strong data encryption technique and trusted third party to maintain the database. The rapid development in information technology, the secure transmission of confidential data herewith gets a great deal of attention. The conventional methods of encryption can only maintain the data security. The information could be accessed by the unauthorized user for malicious purpose.

Therefore, it is necessary to apply effective encryption/decryption methods to enhance data security Strong cryptography or cryptographically strong is general terms applied to cryptographic systems or components that are considered highly resistant to cryptanalysis.

Transmission of sensitive data over the communication channel have emphasized the need for fast and secure digital communication network to achieve the requirements for secrecy, integrity and non-reproduction of exchanged information. Cryptography provides a method for securing and authenticating the transmission of information over secure channels. It enables us to store sensitive information or transmit it across insecure network so that unauthorized persons cannot read it.

Cryptography refers to encryption, the process of converting ordinary information (plaintext) into unintelligible cipher text Decryption is the reverse, moving from unintelligible cipher text to plaintext. A cipher is a pair of algorithms which creates the encryption and the reversing decryption. The detailed operation of a cipher is controlled both by the algorithm and, in each instance; by a key .This is a secret parameter for a specific message exchange context. Keys are important, as ciphers without variable keys are trivially breakable and therefore less than useful for most purposes. Historically, ciphers were often used directly for encryption or decryption,

without additional procedures such as authentication or integrity checks.
Cryptography is the methods that allow information to be sent in a secure from in such a way that the only receiver able to retrieve this information. Presently continuous researches on the new cryptographic algorithms are going on. However, it is a very difficult to find out the specific algorithm, because we have
already known that they must consider many factors like: security, the features of algorithm, the time complexity and space complexity.

II. GRAMMAR

A. Context-Free Grammars

Many cryptographic algorithms use one-way functions to provide their security against adversaries, but still be useful for authorized parties. A one-way function is a function that given x, it is easy to find f(x). However, given f(x) it is hard to find x. An algorithm that uses context free grammars is proposed in this paper

A context-free grammar (CFG) is a set of recursive rewriting rules (or *productions*) used to generate patterns of strings.

A CFG consists of the following components:

- a set of *terminal symbols*, which are the characters of the alphabet that appear in the strings generated by the grammar.
- a set of *nonterminal symbols*, which are placeholders for patterns of terminal symbols that can be generated by the nonterminal symbols.
- a set of *productions*, which are rules for replacing (or rewriting) nonterminal symbols (on the left side of the production) in a string with other nonterminal or terminal symbols (on the right side of the production).
- a *start symbol*, which is a special nonterminal symbol that appears in the initial string generated by the grammar.

Context-free grammars are strictly more powerful than regular expressions.

- Any language that can be generated using regular expressions can be generated by a context-free grammar.
- There are languages that can be generated by a context-free grammar that cannot be generated by any regular expression.

As a corollary, CFGs are strictly more powerful than DFAs and NDFAs.

III. CURRENT SYSTEM

The most straight-forward attack on an encrypted message is simply to attempt to decrypt the message with every possible key. Most of these attempts will fail. But one might work. At which point you can decrypt the message.
Most encryption algorithms can be defeated by using a combination of sophisticated mathematics and computing power. The results are that many encrypted messages can be deciphered without knowing the key. A skilled cryptanalyst can sometimes decipher encrypted text without even knowing the encryption algorithm.
AES is a new cryptographic algorithm that can be used to protect electronic data. Specifically, AES is an iterative, symmetric-key block cipher that can use keys of 128, 192, and 256 bits, and encrypts and decrypts data in blocks of 128 bits (16 bytes). Unlike public-key ciphers, which use a pair of keys, symmetric-key ciphers use the same key to encrypt and decrypt data. Encrypted data returned by block ciphers have the same number of bits that the input data had. Iterative ciphers use a loop structure that repeatedly performs permutations and substitutions of the input data.
AES is the successor to the older Data Encryption Standard (DES).The AES algorithm is based on permutations and substitutions. Permutations are rearrangements of data, and substitutions replace one unit of data with another. AES performs permutations and substitutions using several different techniques
The AES encryption routine begins by copying the 16-byte input array into a 4×4 byte matrix named State. The encryption algorithm performs a preliminary processing step that's called AddRoundKey in the specification.

AddRoundKey performs a byte-by-byte XOR operation on the State matrix using the first four rows of the key schedule, and XORs input State[r,c] with round keys table w[c,r].

The main loop of the AES encryption algorithm performs four different operations on the State matrix, called Sub Bytes, Shift Rows, Mix Columns, and AddRoundKey in the specification. The AddRoundKey operation is the same as the preliminary Add Round Key except that each time AddRoundKey is called; the next four rows of the key schedule are used. The SubBytes routine is a substitution operation that takes each byte in the State matrix and substitutes a new byte determined by the Sbox table. ShiftRows is a permutation operation that rotates bytes in the State matrix to the left. The Mix Columns operation is a substitution operation that is the trickiest part of the AES algorithm to understand. It replaces each byte with the result of mathematical field additions and multiplications of values in the byte's column.

The addition and multiplication are special mathematical field operations, not the usual addition and multiplication on integers.

The four operations SubBytes, ShiftRows, MixColumns, and AddRoundKey are called inside a loop that executes Nr times—the number of rounds for a given key size, less 1. The number of rounds that the encryption algorithm uses is 10, 12, or 14 and depends on whether the seed key size is 128, 192, or 256 bits.

The new AES will certainly become the de facto standard for encrypting all forms of electronic information, replacing DES. AES-encrypted data is unbreakable in the sense that no known cryptanalysis attack can decrypt the AES cipher text without using a brute-force search through all possible 256-bit keys.

IV. PROPOSED SYSTEM

The algorithm makes cryptanalysis even more difficult because of the use of "Random Number Generator" function which further decides order of encryption rounds and keys to be used to encrypt the plain text. This eliminates the overhead of defining a fixed key by the user and makes algorithm secure also. With secret key cryptography, a single key is used for both encryption and decryption. The key selection mechanism and the encoding methodology express the efficiency of the cipher text generated.

Context free grammars present the desirable cryptographic property that it is easy to generate and validate strings from a given grammar; however it is hard to identify a grammar given only the strings generated by it. The project aims at developing a CFG-based cryptosystem that will encrypt a text file to protect from various security attacks.

This cryptosystem will use a symmetric algorithm that will have a secret key. The text file will be converted into a cipher text which will be sent to the receiver who will decrypt it.

```
Procedure Key Generation ()
Input: text
Output: secret key
Begin
Enter text
Generate secret key
End
```

Figure 1: Algorithm for key generation

```
Procedure Encryption ()
Input: plain text file
Output: cipher text
Begin
Key stuffing in plain text file
Reassigning ASCII
Count the number of characters in the
text file
If c mod 16= =0
Generation of matrices
Generate reverse productions
Generate ASCII and binary values
Stuffing of the bits
Else
Stuff space characters
Generate cipher text
End
```

Figure 2: Algorithm for encryption

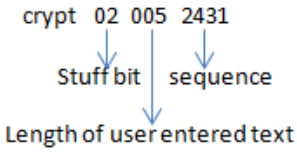
Procedure decryption ()
Input: cipher text, secret key
Output: plain text file
Begin
Generate binary
Unpacking the bits
Generate ASCII values
Generate reverse productions
Reverse even odd method
Generate matrices
Key extraction
Key matching
If secret key = = key in text file

Display plain text
Else
Display garbage value
End

Figure 3: Algorithm for decryption

Considering an example:
Plain text file: this is the text to be encrypted
User entered text: crypt
Generated key (secret key): crypt020052431
The first level is **KEY GENERATION** where the following processes will occur:
-user will enter text
-the secret key will be generated

The secret key is divided into 4 parts



02 in the secret key indicates after how many places to stuff the key, 005 is the length of the user entered text, 2431 is he matrix sequence which is randomly generated.

The next level is **ENCODE** where the following processes will occur:
-The user will input the text file
-Stuffing of the key into the file
-Reassigning ASCII values
-count the number of characters in the text file
-generate 4 X 4 matrices

After stuffing the key we get:
Thcisr iys pa ttext to be encrypted

The algorithm reassigns the ASCII values to generate the following string:
1% &0/<&60<-}<11”51<1,~”<”+ /6-1”!

It will then count the number of characters in the text file
Let ‘c’ be the count
c = 34
Now compute c mod 16
34 mod 16 = 2
16-2=14
Thus, 14 space characters will be stuffed in the text file

The next step is to generate the 4 x 4 matrices:

2	4	3	1
1	%				
0	/	<	&
6	0	<	-
}	<	1	1

2	4	3	1
“	5	1	<
1	,	~	“
<	“	+		
/	6	−	1

2	4	3	1
“	!	<	<
<	<	<	<
<	<	<	<
<	<	<	<

(Note: “<“ is used to indicate the stuffed space characters in the above matrices just for understanding)

Shuffling of matrices:

2	4	3	1
&	&	-	1
1	0	6	}
	<	<	1
%	/	0	<

2	4	3	1
<	“		1
“	1	<	/
1	~	+	-
5	/	“	6

2	4	3	1
<	<	<	<
“	<	<	<
<	<	<	<
!	<	<	<

Hence we get the string:
1}1<&1
%-6<0&0</1/-6<”15<+””1~/<<<<<”<!<<<<<<<<

The third level is the **ENCRYPT** level which begins by generating reverse productions using context free grammar (CFG)

1}1<&1 % **-6<0&0</** **1/-6<”15**
A1->%1 A2 } A4->/- A5 <6 A7->51 A8 1/
A2->11 A3 &< A5->0< A6 &0 A8->”-A9<6
A3->E A6->E A9->E

<+””1~/ **<<<<<”<!** **<<<<<<<<**
A10->/ A11~< A13->!<A14<< A16->><<A17<<
A11->1+ A12 “” A14->”< A15<< A17->>< A18<<
A12->E A15->E A18->E

After eliminating the non-terminals we get the string:
%1
}11&</-<60<&0511/”-<6/~<1+””!<<<”<<<<<
<<<<<<

The algorithm then generates the ASCII and binary values of each character in the text file.

After stuffing of the bits we get:
01100010010000011111101001100010011000
11010011000111100101011010011110010110
11000110000101111001010011010110000101
10001001100011010111110100010010110110
11110000110110101000000111111010111100
00110001101010111010001010100010101111
00001111001011110000100010001111000011
11000011110010111100001111001011110010
11110010111100101111000011110000

This file is then converted to the cipher text:
bAúbcLyZylayMabc_D¶ðÚ úðÆ®ŠŠðòðˆððò
ðòòòòð

This cipher text file will then be sent to the receiver who will decrypt it using the same secret key.
The decryption process consists of the **DECRYPT** and the **DECODE** level which is exactly opposite to encryption except that key matching will occur at the end of the algorithm. If the key is matched only then the plain text will be displayed to the receiver.

V. DESIGN

The flowcharts explain the flow of the algorithm and its different levels of processing, both for encryption and decryption.

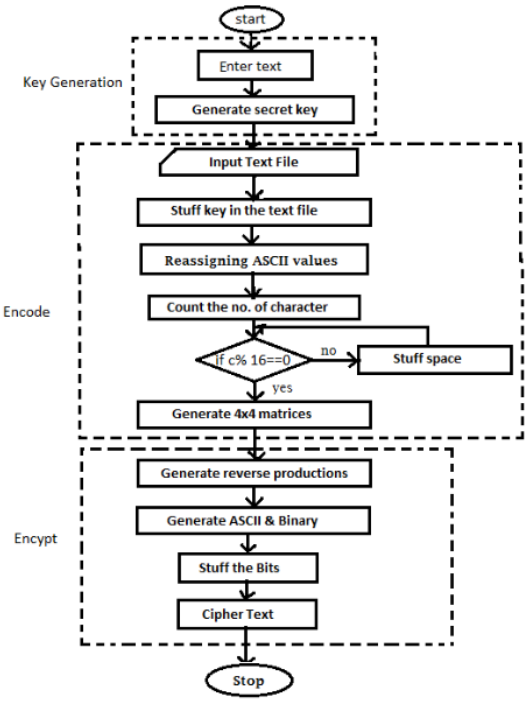


Figure 4: Flowchart for Encryption

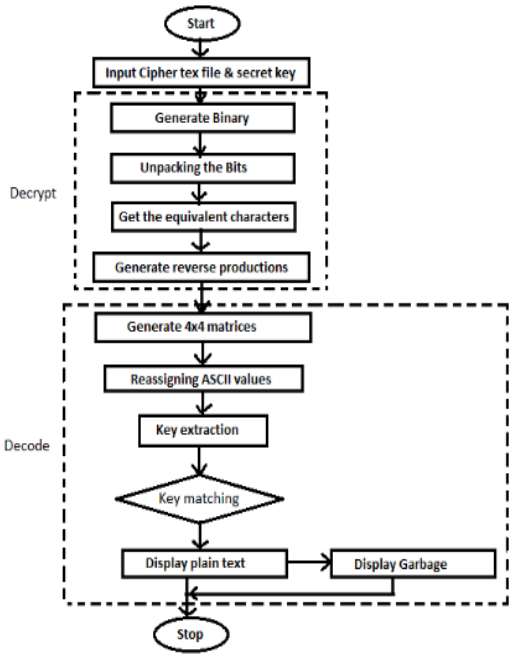


Figure 5: Flowchart of Decryption

VI. SECURITY ATTACKS

Brute force

In cryptography, a brute-force attack, or exhaustive key search, is a cryptanalytic attack that can, in theory, be used against any encrypted data. Such an attack is not possible on our algorithm because of two reasons firstly, since we are using context free grammar it is very easy

to generate productions but very difficult to get back the grammar. Secondly, we are using 128 bit key so it will require 2^{128} combinations which is a very big number for the attacker to try.

Cryptanalysis

Most encryption algorithms can be defeated by using a combination of sophisticated mathematics and computing power. The results are that many encrypted messages can be deciphered without knowing the key. A skilled cryptanalyst can sometimes decipher encrypted text without even knowing the encryption algorithm.

A cryptanalytic attack can have two possible goals. The cryptanalyst might have ciphertext and want to discover the plaintext, or the cryptanalyst might have ciphertext and want to discover the encryption key that was used to encrypt the message. The following attacks are commonly used when the encryption algorithm is known. These may be applied to encrypted files or Internet traffic:

Known Plaintext Attack

The goal of a known plaintext attack is to determine the cryptographic key and possibly the algorithm which can then be used to decrypt other messages. Since we are using random key generation algorithm every time the key will be different therefore the attacker won't be able to determine the key.

VII. CONCLUSION

A powerful cryptosystem has been proposed in this paper. The protocol discussed provides security without requiring additional layer of encryption like SSL. The protocol's use of context free grammar provides its security.

The salient features of the algorithm include no large overheads, user friendliness and independent of any other cryptographic protocol. The described cryptosystem makes use of interesting issues regarding context free grammars that until now have only been used to design programming languages. It's desirable cryptographic property that it is easy to generate and validate strings from a given grammar but it is hard to identify the grammar given only the strings generated by it. The proposed system does not rely on any other cryptographic protocols. This paper presents and analyzes the protocol with respect to its robustness against malicious attacks.

ACKNOWLEDGMENT

The authors are grateful to the anonymous reviewers for their valuable advices. The authors also wish to express their thankfulness to Prof. Manisha G. Fal Dessai and Prof. Amey Kerkar for their invaluable suggestions.

REFERENCES

[1] Abhishek Singh, Andre L M dos Santos, "*Context Free Grammar for the Generation of a One Time Authentication Identity*"

[2] Abhishek Singh, Andre L M dos Santos, "*Grammar Based Off line Generation of Disposable Credit Card Numbers"*

TEXTBOOK

1. Cryptography and Network security 4th ed. William Stallings PEA, ISBN:978-81-7758-774-6.