European Journal of Scientific Research
ISSN 1450-216X Vol.78 No.4 (2012), pp.546-545

http://www.europeanjournalofscientificresearch.com

# Proposed New Algorithm to Generate Cryptography Session Keys Based on CFG and Huffman Code

**Hala Bahjat AbdulWahab**
*Computer Science Department, University of Technology*
*Baghdad, Iraq*
E-mail: hala_bahjat@yahoo.com

**Alia Karim AbdulHassan**
*Computer Science Department, University of Technology*
*Baghdad-Iraq*
E-mail: hassanalia2000@yahoo.com

**Nedhal A. Al-Saiyd**
*Computer Science Department, Faculty of Information Technology*
*Applied Science University, Amman, Jordan*
E-mail: nedhal_alsaiyd@asu.edu.jo

**Abstract**

The key management is an important area of research in internet applications, because protecting secret messages during transmission becomes an essential issue for the Internet. The PGP cryptography protocol work as first stage to creates a session key, which is a one-time-only secret key. This key is a random number generated from the random movements for mouse and the keystrokes when the person type. This paper proposes new algorithm to generate session keys instead of the traditional method that used in PGP protocol stages. The proposed algorithm based on Context Free Grammar to build meaningful sentences with related production rules. The CFG production rules are coupled with Huffman coding and used as a tool for flexible generating of session keys and to increase the security of the system.

**Keywords:** Session Keys, PGP Cryptography Protocol, Formal Languages, Context Free Grammar, CFG Production Rules, Huffman Coding

## 1. Introduction

*Cryptography* is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication, Cryptography is not the only means of providing information security, but rather one set of techniques [1]. It is no surprise, then, that new forms of cryptography came soon after the widespread development of computer communications, and the development of Internet technologies. In data and telecommunications, cryptography is necessary when communicating over any entrusted medium, which includes as authentication and privacy conditions in any network, particularly the Internet [2].

A session key is an encryption and decryption key that is randomly generated to ensure the security of a communications session between a user and another computer or between two computers.

Session keys are sometimes called *symmetric key*s, because the same key is used for both encryption and decryption. A session key may be derived from a hash value; using the *CryptDeriveKey* function (this method is called a *session-key derivation scheme*). Throughout each session, the key is transmitted along with each message and is encrypted with the recipient's public key. Because much of their security relies upon the brevity of their use, session keys are changed frequently. A different session key may be used for each message.[3]

A cryptographic protocol is a set sequence of messages exchanged between two or more parties who are trying to accomplish something(i.e. the term "something" is what we call the protocol's goal – perhaps it is the ability to establish a shared secret or maybe it is just the reassurance that a certain someone is still alive. Whatever the goal, the ability to accomplish it, despite the presence of some adversary bent on being evil, is what we call protocol security [4]. A cryptographic scheme for a given task is secure if no adversary of a specified power can achieve a specified break [5].

In this work, both symmetric and asymmetric cryptographic mechanisms are used together in order to take advantage of the strengths of each; specifically, the ability to distribute public keys without concerns of confidentiality (a strength of asymmetric key cryptography), and the faster encryption/decryption speed of secret keys (a strength of symmetric key cryptography relative to asymmetric). Cryptography is not the only means of providing information security, but rather one set of techniques such as use context free grammar to build meaningful sentences with related production rules. The CFG production rules coupled with Huffman coding as a tool for flexibility for key exchange that increase the security of the system.

## 2. Key Management

One of the major roles of public-key encryption has been to address the problem of key distribution. There are actually two distinct aspects to the use of public-key cryptography in this regard [3]:

2.1. The distribution of public keys
2.2. The use of public-key encryption to distribute secret keys

### 2.1. The Distribution of Public Keys

Several techniques have been proposed for the distribution of public keys. Virtually all these proposals can be grouped into the following general schemes:

- Public announcement
- Publicly available directory
- Public-key authority
- Public-key certificates

### 2.2. Distribution of Secret Keys Using Public-Key Cryptography

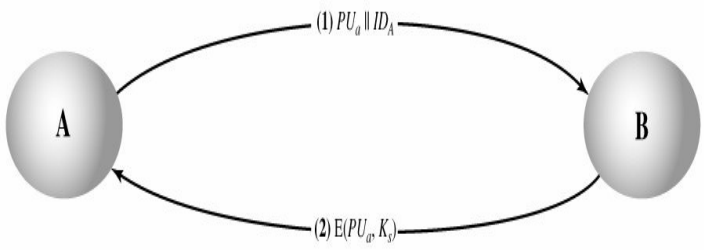
Once public keys have been distributed or have become accessible, secure communication that thwarts eavesdropping, tampering, or both is possible. However, few users will wish to make exclusive use of public-key encryption for communication because of the relatively slow data rates that can be achieved. Accordingly, public-key encryption provides for the distribution of secret keys to be used for conventional encryption.

#### 2.2.1. Simple Secret Key Distribution

An extremely simple scheme was put forward by Merkle [3], as illustrated in Figure (1). A and B can now securely communicate using conventional encryption and the session key $K_s$. At the completion of the exchange, both A and B discard $K_s$. Despite its simplicity, this is an attractive protocol. No keys exist before the start of the communication and none exist after the completion of communication.

Thus, the risk of compromise of the keys is minimal. At the same time, the communication is secure from eavesdropping.

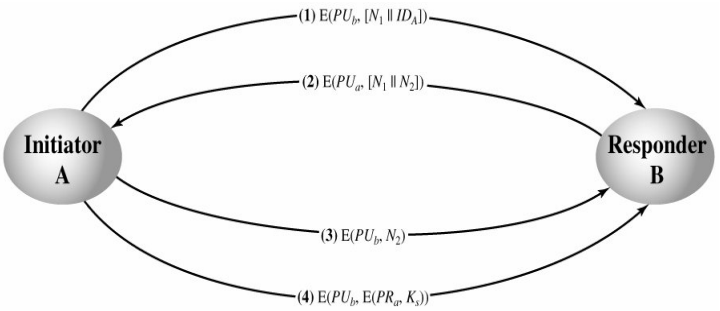
**Figure 1:** Simple Use of Public-Key Encryption to Establish a Session Key




### 2.2.2. Secret Key Distribution with Confidentiality and Authentication

Figure (2), based on an approach suggested in [3], and provides protection against both active and passive attacks. We begin at a point when it is assumed that A and B have exchanged public keys by one of the schemes described earlier in this section.

**Figure 2:** Public-Key Distribution of Secret Keys




Notice that the first three steps of this scheme are the same as the last three steps of Figure (2). The result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.

### 2.2.3. A Hybrid Scheme

Yet another way to use public-key encryption to distribute secret keys is a hybrid approach in use on IBM mainframes. This scheme retains the use of a key distribution center (KDC) that shares a secret master key with each user and distributes secret session keys encrypted with the master key. A public key scheme is used to distribute the master keys. The following rationale is provided for using this three-level approach:

- Performance: There are many applications, especially transaction-oriented applications, in which the session keys change frequently. Distribution of session keys by public-key encryption could degrade overall system performance because of the relatively high computational load of public-key encryption and decryption. With a three-level hierarchy, public-key encryption is used only occasionally to update the master key between a user and the KDC.
- Backward compatibility: The hybrid scheme is easily overlaid on an existing KDC scheme, with minimal disruption or software changes.

The addition of a public-key layer provides a secure, efficient means of distributing master keys. This is an advantage in a configuration in which a single KDC serves a widely distributed set of users.

In this paper, we produce a new algorithm using a hybrid scheme with new tools, consists the CFG and Huffman code techniques.

## 3. Context Free Grammar (CFG)

Grammars are designed to describe languages, where in our context a language is just a set of strings. Abstractly, we think of strings as a sequence of so-called terminal symbols.

A context-free grammar consists of a set of productions of the form $X \rightarrow \gamma$ , where X is a non-terminal symbol and is a potentially mixed sequence of terminal and non-terminal symbols. It is also sometimes convenient to distinguish a start symbol traditionally named S, for sentence. We will use the word string to refer to any sequence of terminal and non-terminal symbols. We denote strings by $\alpha, \beta, \gamma, \ldots$. Non-terminals are generally denoted by X, Y, Z and terminals by a, b, c.

For example, the following grammar generates all strings consisting of matching parentheses.

$$\begin{aligned} &S \rightarrow \\ &\quad S \rightarrow [S] \\ &\quad S \rightarrow SS \end{aligned} \tag{1}$$

A *derivation* of a sentence w from start symbol S is a sequence $S = \alpha_0 \rightarrow \alpha_1 \rightarrow \alpha_n = w$, where w consists of only terminal symbols. In each step, we choose an occurrence of a non-terminal X in αi and a production $X \rightarrow \gamma$ and replace the occurrence of X in αi by $\gamma$ .

We usually label the productions in the grammar so that we can refer to them by name. In the example above, we might write:

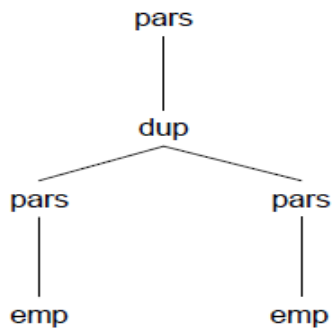
$$\begin{aligned} &[emp]\ S \rightarrow \\ &\quad [pars]\ S \rightarrow [S] \\ &\quad [dup]\ S \rightarrow SS \end{aligned} \tag{2}$$

Then the following is a derivation of the string [7], where each transition is labeled with the production that has been applied.

$$\begin{aligned} S &\rightarrow [S] && [pars] \\ &\rightarrow [SS] && [dup] \\ &\rightarrow [[S]S] && [pars] \\ &\rightarrow [[]S] && [emp] \\ &\rightarrow [[][S]] && [pars] \\ &\rightarrow [[][]] && [emp] \end{aligned} \tag{3}$$

We have labeled each derivation step with the corresponding grammar production that was used. Derivations are clearly not unique, because when there is more than one non-terminal we can replace it in any order in the string. In order to avoid this kind of harmless ambiguity in rule order, we like to construct a parse tree in which the nodes represents the non-terminals in a string, with the root being S. In the example above we obtain the following tree in figure(3):

**Figure 3:** The Parsing Process.




While the tree removes some ambiguity, it turns out the sample grammar is ambiguous in another way. In fact, there are infinitely many parse trees of every string in the language. This can be seen by considering the cycle

$$S \rightarrow SS \rightarrow S \quad (4)$$

Where the first step is dup and the second is **emp,** applied either to the first or second occurrence of S [8].

## 4. Huffman Encoding Technique

Huffman coding is a method for instantaneous encoding schemes. This method starts by building a list of particular alphabetical symbols in descending order of their probabilities. It then constructs a tree, with a symbol at every leaf, from the bottom up tree. This is done in ordered steps, where at each step the two alphabetical symbols with the smallest probabilities are selected, added to the top of partials tree, deleted from the list and replaced with an auxiliary symbol representing both of the alphabetical symbols. When the list is reduced to just one auxiliary symbol (representing the entire alphabet) the tree is complete [6].

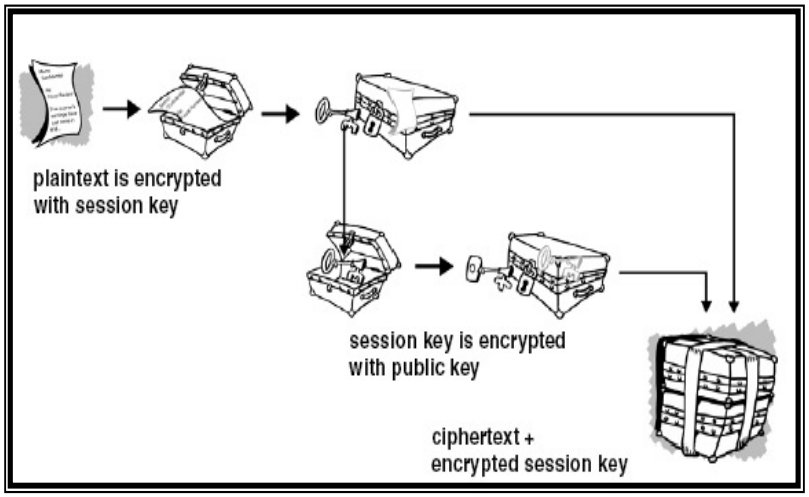
## 5. PGP as A Hybrid Cryptosystem

Pretty Good Privacy (PGP) is a public key system for encrypting electronic mail using the RSA public key cipher [8]. PGP combines some of the best features of both conventional and public-key cryptography. PGP is a hybrid cryptosystem.

When a user encrypts plaintext with PGP, PGP first compresses the plaintext. Data compression saves modem transmission time and disk space and, more importantly, strengthens cryptographic security. Most cryptanalysis techniques exploit patterns found in the plaintext to crack the cipher. Compression reduces these patterns in the plaintext, thereby greatly enhancing resistance to cryptanalysis. (Files that are too short to compress or which do not compress well are not compressed.)

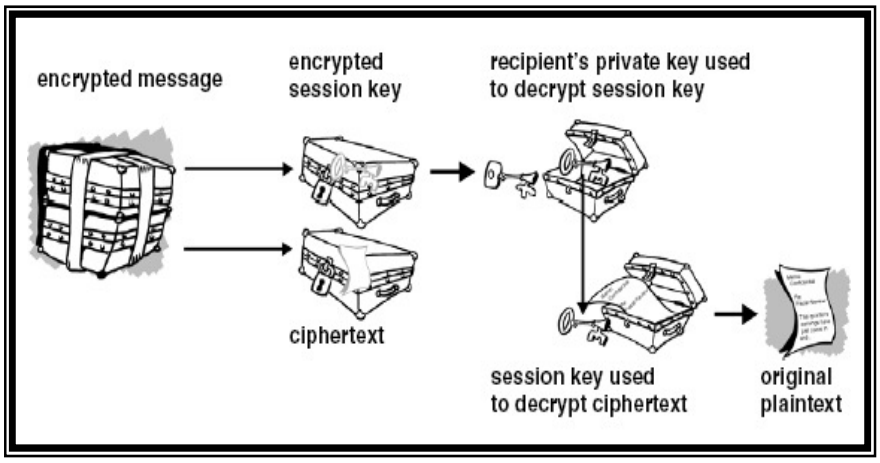
PGP then creates a session key, which is a one-time-only secret key. This key is a random number generated from the random movements of your mouse and the keystrokes you type. The session key works with a very secure, fast conventional encryption algorithm to encrypt the plaintext; the result is ciphertext. Once the data is encrypted, the session key is then encrypted to the recipient's public key. This public key-encrypted session key is transmitted along with the ciphertext to the recipient. Figure (4) shows the send process.

**Figure 4:** Send Process.




Decryption works in the reverse. The recipient's copy of PGP uses his or her private key to recover the session key, which PGP then uses to decrypt the conventionally encrypted ciphertext. In figure (5) shown the received process.

**Figure 5:** Received Process

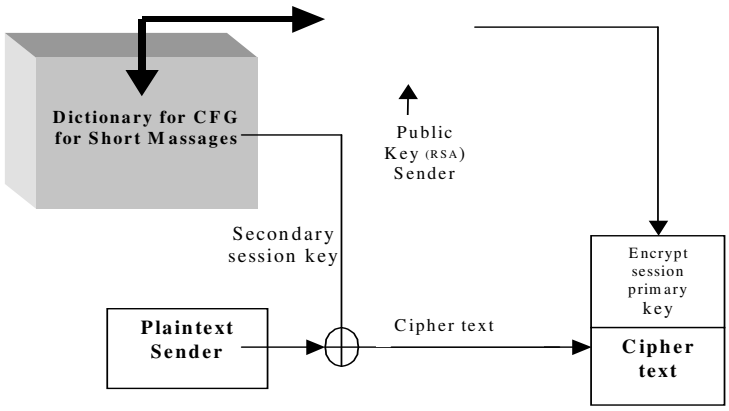



The combination of the two-encryption methods combines the convenience of public-key encryption with the speed of conventional encryption. Conventional encryption is about 10,000 times faster than public-key encryption. Public-key encryption in turn provides a solution to key distribution and data transmission issues. Used together, performance and key distribution are improved without any sacrifice in security.

## 6. The Proposed Algorithm to Generate Session Keys

The proposed algorithm to generate session key used to modify the PGP Cryptography cryptosystem. The strong cryptography employed by PGP is the best available today. The PGP protocol is a hybrid cryptosystem that combines some of the best features of both conventional and public-key cryptography figure (6) show the general view of the proposed PGP cryptosystem.

**Figure 6:** Proposed PGP Cryptosystem




In this propose insert the new algorithm for generate the session keys to the PGP protocol stages to increase protocol robustness and make the protocol more difficult in front of the counterfeiter. This paper aim used the CFG and Huffman code facilities to generate random a session key instead of generating the session key by using movement of the mouse or the keystrokes type. The session key that generated according the proposed algorithm consist of two session keys, *primary session key* and *secondary session key*.

Primary session key represent the grammar that derivation from short massage and secondary session key represent the stream of randomness bits sequence that is generated according irreducible polynomials to increase the randomness bits sequence. The works with New-PGP begin when a user encrypts plaintext. *First,* compress the plaintext . *Second,* creates a session key select from the dictionary the short massage. *Third,* the user enter primary session key( initial value to the LFSR) in order generate secondary session key ( sequences randomness bits). *Forth*, the user XOR the stream of the secondary session key bits sequence with the plaintext after compression process. *Fifth*, the sender uses the public key from RSA algorithm to encrypt the primary session key. *Sixth,* the sender transmits the encrypted primary session key along with the cipher text to the recipient. Decryption works in the reverse order. The recipient's copy of new-PGP uses his or her private key from RSA algorithm to recover the primary session key that is used to generate the secondary key from generated image to decrypt the conventionally encrypted cipher text. The detailed algorithm for the proposed described in algorithm1.

**Algorithm 1:** Generate Session Cryptographic Keys Using CFG and Huffman Code

| | |
|---|---|
| **Input**: | Message M (each line of message is $M_i$) |
| **Output:** | Cryptographic Session keys |
| **Process**: | |
| **Step1** | Build dictionary from short massages that agreed between the sender and Receiver.( this dictionary more flexible to update like add, delete, …).<br>Choose a short message from dictionary (message length variable between single characters to number of lines)<br>Divide the message m to set of sub sentence( n ) and use a mark at each one. |
| **Step 3** | Construct the CFG grammar for m |
| **Step 4** | Add code to the grammar using Huffman code.( code length is equal to number of the sub sentences( for example let n=3bit ) |
| **Step 5** | Parse path according CFG |
| **Step 6** | Compression (skip the 0's at left hand side) |
| **Step 7** | Find the polynomial according the result that obtained from step-6. |
| **Step 8** | Select predetermined irreducible polynomial degree. |
| **Step 9** | Construct LFSR using irreducible polynomial that select in step-8. |
| **Step 10** | LFSR generate the random bit sequence (Session key sequence). |
| **Step 11** | Test the generated random bit sequence using 5-tests.<br>End. |

**Example 1**

Each step in algorithm1 will be described in the following example:

Step1: Choose a short message (message length variable between single characters to number of lines)

**Meeting :9 o'clock at my home**

Step2: Divide the message M according for the length for selected massage to set of sub sentence( n ) and use a mark at each one.

M={ meeting:9#, o'clock@, at my home.}

Step 3: Construct the CFG grammar for M

**S→meeting:9#**

**A→o'clock@**

**B→at my home.**

Step 4: Add code to the grammar using Huffman code.( code length is equal to number of the sub sentences( n=3bit )

**S→meeting:9# , 000**
**A→o'clock@ , 010**
**B→at my home. , 011**

Step 5: Parse path according CFG
**000 010 011**

Step 6: Compression process: (skip the 0's at left hand side) in order find the degree for the polynomial =>> **10011=>> find equvilant polynomail= $f(x)=1+x^3+x^4$**

Step 7: Select predetermined irreducible polynomial degree greater than the n→ 5 For this example $F(x)=1+x^3+x^5$.

Step 8 : Construct LFSR from $F(x)=1+x^3+x^5$ , apply the seed that represent compression code = 10011 ( that represent the master session key).

Step 9 : LFSR generate the random bit sequence (key sequence with period $2^n$-1=15 bits)➔ k= 111101101101101 10

Step 10: Test the generated random bit sequence using five popular randomness test.

Step 11: Convert the message that Alice want to sent it to Bob
M= "TO BE OR NOT TO BE"➔ASCII code ➔
84 79 66 69 79 82 78 79 84 79 66 69
Let " TO" ➔ binary form =1010100 1001111.

Step 12: Xor key sequence with message sequence to produce cipher sequence
$K_i \oplus M_i \rightarrow C_i$ ➔ (111101101101101 10) ⊕ (1010100 1001111)=
(01110010010100) cipher text for two character "TO".

Step 13 Send $C_i$ Using PGP protocol stages.

Step 14 End.

**Example 2**

To explain how the New-Protocol works, and indicate the protocol behavior. used the secret massage binary form as primary session key (PSK) that consists of select short massage with binary form . According to the primary session key we generate the secondary session key (SSK) of size equal to 15 random bits using LFSR according irreducible polynomials. The public key (PK) of the RSA algorithm consist of (n=997517,e=193) where (secret p =977) and (secret q=1021), and the private key of RSA algorithm equal (d=727297).

## 7. Conclusions

Every few years, it is necessary for computer security to re-invent itself. New technologies and new applications bring new threats and, with the ever-increasing growth of data communication, the need for security and privacy has become a necessity. Cryptography and data security are an essential requirement for communication privacy. The main goal of this paper was to combine the CFG methods with cryptography algorithms in order to increase the capability of cryptography. The weakness of the cryptographic key generated from a traditional methods is clear . This led to the proposal in this paper of a new method in order to generate a session cryptographic key depending on CFG.

From the New-PGP method, reached to the following conclusions:-

a. The process to guesses the primary session key from the secondary key is infeasible, because there is no correlation between the two session keys.
b. If the counterfeiter succeed to solve the factorization problem from RSA , and find the private key from public key , the key that is obtained cannot help him to recover the plaintext from primary session key unless knowing the secondary session key.

c. The New-PGP increased secure condition to the PGP protocol that made the protocol more robust and efficient.

## References

[1] M. P. van Oorschot, and S. Vanstone, "*Handbook of Applied Cryptography*", CRC Press, 1996.
[2] http://www.garykessder.net/library/crypto.html(may1998-17 January 2007 ), "*an overview of cryptography*".
[3] *William Stalings, "Cryptography and Network Security (Principles and Practice)", Pearson Education, Inc,. (2005).*
[4] A. Patil, "*On Symbolic Analysis of Cryptographic Protocols*", Master of Engineering in Computer Science and Engineering, at the MIT, May 2005.
[5] J. Katz, Y. Lindell, "*Introduction to Modern Cryptography*", Chapman & Hall/CRC, 2008.
[6] F. Pfenning, "*Lecture Notes on Context-Free Grammars 15-41*", Compiler Design Lecture 7 ,September 15, 2009.
[7] "http://www.pgp.com, "An Introduction to Cryptography ", 8.0.2. Released May 2003.
[8] Schaefer, E. D. "***An introduction to Cryptography***", Santa Clara University,1999