SMCC Higher Education Research Journal
ISSN Print: 2449-4402 · Volume 1 · January 2015

Hybrid Encryption Algorithm Implementation on Electronic Mail Service

CRISELDO C. CALINAWAN
http://orcid.org/0000-0001-8208-8082
calinawancriseldoc@smccnasipit.edu.ph
College of Computer Studies
Saint Michael College of Caraga
Atupan St., Nasipit, Agusan del Norte, Philippines

ABSTRACT

The study focuses on two main objectives, to develop a Hybrid Encryption Algorithm and design a Pretty Good Privacy using the developed hybrid encryption algorithm. The researcher developed hybrid encryption algorithm composed of two known and effective encryption algorithms namely the RSA and Solitaire Encryption Algorithms. The goal is to enhance its security vulnerability against any unexpected attacks, reliability and speed of execution. Utilizing a Bluff key generation where it hides the session key used to generate the keystream. Eventually decreased the vulnerability of the algorithm from hackers or cyber attackers. The developed algorithm will then be integrated on an existing open source email client. As a result, the system, as expected, will provide a new breed of Pretty Good Privacy system. The sender will be able to send and receive encrypted email in text format messages over the internet. The system is designed to be more robust and flexible compared to other existing email clients with PGP feature. That even a non-PGP email services online will be able to decrypt the email messages in a user-friendly interface. An RSA key management protocol is utilized to decipher the message confidentially using a private key that is available for download online. The proponent utilized the existing open source Roundcube

Webmail client. The result showed that the algorithm executes faster compared to the existing RSA algorithm at an average time difference of 0.11 seconds using similar hardware specifications and software requirements. The developed PGP System has a particular limit on formatted and plaintext encryption/decryption processes where it does not support Multipurpose Internet Mail Extensions (MIME) handling.

KEYWORDS

Computer science, information technology, hybrid encryption algorithm, pretty good privacy (pgp), email, open source email client, encryption, decryption, bluff key, R&D Model, Agusan del Norte, Philippines, Asia

INTRODUCTION

William, S., & Stallings, W. (2006) cited that security threats are constantly an issue when it comes to electronic communications. And for email applications, Gaboitsiwe (2013) stresses that the security threat is decisive when it comes to confidential messages sent over the internet. Since the email messages run through a hefty number of networks all over the world, the risk of intrusion increases. More computer programs emerge from within, where it is capable of hiding information from the view of the infiltrators.

Data encryption is one of the most outstanding data security implementation. It converts readable information into an unrecognized one where the user can only interpret thru decryption process (Singh, 2011). Copiously of them are in existence and widely use on "secured" systems of business firms, government offices, and others.

Schneier (2007) pointed out that it is undeniable that encryption algorithms can be crack down after several years without enhancing it, most especially when the system expose to the public view and even taught in computer classes. And it is true that some computer experts even geniuses tried to crack down top encryption algorithm just to test their limitations or for personal satisfaction.

Encryption algorithms do have setbacks, and that is one of the reasons why plenty of different algorithm standards emerge, that include the following factors: speed performance, reliable, complexity and bit size. With these factors, developers tried to enhance or even develop a new one just to compensate what the older version don't have.

This study focuses on the development of new encryption algorithm based on existing algorithms. The algorithm provides us with enhanced encryption and decryption algorithm that makes it immune to infiltrators; that eliminates some of the factors that affect the performance of an encryption algorithm. Also, the algorithm will be implemented in an online electronic mail service as its primary application that resembles more likely the same idea with Pretty Good Privacy (PGP).

FRAMEWORK

An Analysis of Key Management in Cryptographic Algorithms

The study focused and discussed on the distribution of key values in the secret way either using any centralized key distribution server or self-organized key management in public networks based on Ramaraj, Karthikeyan & Laitha (2006) approach.

Key distribution server

This study focuses on several solutions for secure key exchange between the recipient and the sender using a centralized key distribution server. In this protocol, the sender requires a session key for communicating the receiver. First, the sender communicates the key distribution server or known as KDC (Key Distribution Center) and encrypts to copies of randomly generated session key. When I encrypt one copy for the sender's public key, another copy is encrypted using the recipient's public key. And then KDC sends the two encrypted keys to the sender. The sender then decrypts one random key using his secret key and then identifies the session key and sends another one copy of encrypted random session key to the recipient. And then the recipient decrypts the session key using his private key. Another method is when the sender wants to send the message to his desired recipient, the sender asks for the public key of the recipient and receives it from KDC. The sender then randomly chooses any session key and encrypts it using recipient's public key. The sender sends the encrypted session key and decrypts it using his private key for this reason both the sender and the recipient knows the session key (Jivsov, 2005). With this protocol, a high risk of intrusion is possible. When recipient sends his public key to the sender, a third party will intercept it and change with the third party's public key and send it to the sender's end. As a result, the intermediate person will be able to decrypt the message. So to avoid this attack, Ron Rivest, and Adi Shamir developed an interlock protocol (Ramaraj, Karthikeyan & Laitha N., 2006).

These interlock protocols mainly divide the message in half. First, the sender sends his public key to a recipient and vice versa. Then sender encrypts the message using the recipient's public key and sends half of the message to the recipient and vice versa. Then the sender sends the other encrypted half of the message to the recipient and the recipient then combines the other half sent first and used recipient's private key to decrypt the entire message. Then recipient's remaining half of the message will be sent to the sender's end and do the same process of retrieving the message. With this protocol, the middle man finds it difficult to alter the message, and he must invent a new message totally and send half of it either to the recipient or the sender. PGP (Pretty Good Privacy) applications use a hybrid encryption system that combines both the conventional and public key cryptography (Park, J. S., & Sandhu, R., 2000). The sender encrypts the message or known as plaintext using PGP, it first compresses the plaintext and creates a session key that is a one-time-only secret key. For this reason session, key works with a very secure and fast conventional encryption algorithm for plaintext encryption.

EAP and Certificate based authentication

In this protocol, to be able to maintain the security of public key's integrity, it should be published with a certificate. The certificate is a data structure that is digitally signed by CA (Certificate Authority). It contains a series of values such as certificate name and usage, owner's public key, the expiration time and CA's name (Bolhuis, 2014).

Self-organized key management

This protocol allows users to generate their public and private key without having centralized authentication servers as well as never use any trusted authority. User's public key must be available to other users in such a way that the authenticity is verifiable that leads to the main problem of public-key based security systems. A user must create a certificate to avoid problems, a data structure in which a public key is bounded to an identity by the digital signature of the issuer. Unlike the latter's protocol, these certificates are stored and distributed in a fully self-organized manner. Key Authentication performs via a chain of public key certificates. When a user wants to obtain a public key of the other user, a chain of valid public key certificates then acquired. Such that the initial certificate of the chain is directly verified by the user, by using a public key that he held and trusted (Tamboli A.S., Shinde S., & Tamboli S.S.,

2011). Each remaining certificate will be verified using public key, contained in the previous certificates of the chain. Then the last certificate contains the public key of the target user. Each node should maintain two local certificate repositories for finding the certificate chains to a different user. Non-updated certificate repository contains expired certificate in which node does not update. The updated certificate repository of a node contains a subset of certificates that is kept updated. The node requests the certificate updates located in the repository when or before they expire (Tamboli A.S., et al., 2011). When the user wants to authenticate the public key of a specific user, both networks merge their updated repositories and user locates the certificate chain of the target user in the merged repository. If found, to authenticate the other user, the user tries to find the other user's certificate chain from the updated and non-updated repositories. To complete the authentication the following steps are followed: User creates its own public and private key pair and then issues his public key certificate to other users based on the security association about other users. After the user identifies the other user, the node then performs certificate exchange.

New Type of Network Security Protocol using Hybrid Encryption in Virtual Private Networking

This study focuses on performing a new type of encryption technique in EAP and MPPE or (Extensible Authentication Protocol and Microsoft-Point-to-Point Encryption respectively). For Hybrid Encryption implementation, the Virtual Private Network server and the VPN client agree on a key, either dynamic or static with 128 bits in length. Then it is used to encrypt the plaintext using AES-Rijndael stream cipher encryption algorithm while the key is encrypted using an RSA algorithm with the recipient's public key and send to the recipient simultaneously which combined. Then VPN client can apply his private key to decrypt the key and use AES-Rijndael to decrypt the message. The main advantage of this method is faster than the existing normal encryption with a secure key transformation (Ramaraj & Karthikeyan, 2006).

The security of the proposed schemes on the difficulty of cryptographic assumptions as follows:

1. EAS-Rijndael algorithm has high-security part compared to other encryption algorithms such as RC4, RC5, etc. (Rodríguez-Henríquez, F., Saqib, N. A., Perez, A. D., & Koc, C. K., 2007).

2. The RSA has a very high security when it comes to public key cryptographic algorithms, but slows when dealing with large numbers of data. So the remedy for this is to encrypt only the key values (Lamprecht, 2012).

The study presented a new type of hybrid encryption protocol for Virtual Private Networks data encryption and key management. The algorithm is more secured compared to the previous algorithms.

OBJECTIVES OF THE STUDY

The researcher developed an existing open source Webmail client. Test the Webmail client and performed encryption when sending email messages to other email servers and received encrypted mail and decrypt it using key management protocol. Existing email servers decrypt the received encrypted email message securely; utilize the use of key management protocol for encryption and decryption

METHODOLOGY

Overview

The development of the new-fangled hybrid encryption algorithm based on the two prevailing effective encryption algorithm to date, which are the RSA and Solitaire encryption algorithms. I know RSA encryption algorithm for its effectiveness on key management (Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A., 1996). It uses the idea of an asymmetric key algorithm that utilize the use of a public and private key in which, in fact, it is best suited for email applications. On the other hand, a Solitaire encryption algorithm is efficient in terms of processing speed and a high level of security (Tounsi, W., Justus, B., Boulahia, N. C., Cuppen, F., & Alfaro, J. G., 2014). Also, an additional bluff key generation is introduced to keep the session key more secured in the eyes of cryptanalyst, which hides the key.

The Hybrid Encryption Algorithm

The algorithm needs two inputs: the *session key* that can be a character, a number, or a combination. The session key is then used to generate a *bluff key*. Then used by the system to generate the *keystream* that is use to encrypt or decrypt the *plain text* (which is provided by the user).

In usual encryption algorithms, after generating a unique key (in our case the session key), the encryption process will take place. Now, one of the differences of this from the existing one is that this algorithm will generate a bluff key to enhancing securely the protection of the session key in which only the sender knows about it. Remember that the strongest point and the most important thing in encryption are the session key especially for symmetric ciphers. With the bluff key, it technically hides the identity of the session key from cryptanalysts. An enhanced way of making the session key immune to intruders, the bluff key will be used to generate the keystream. Then used to encrypt the plain text using mathematical functions used by existing encryption algorithms.

The Bluff Key Algorithm

The design of the bluff key is intended to compress the string length and encrypt the secret key before it is used to generate the keystream. The session key will turn into a bluff key and hides the complete string of the user's session key. An event of attacks using the four methods of a cryptanalysis secures the key while the bluff key appears after attackers can identify the keystream. The bluff key is also designed to be irreversible. I derived the session key from the bluff key, so it does not get decrypted. Below is the bluff generation algorithm.

Let $\boldsymbol{k_i}$ be the secret key and ***Bi*** for the bluff key.

Sample input session key: "secret."

First, convert the characters (session key) to ASCII code in decimal number system,

ki: 115 101 99 114 101 116

Then, let hi be the first value equivalent to the more session key's string length average and its first ASCII character.

r = a randomly generated string in decimal that ranges from 0 to 16.

hi = (Kavg xor ki) +r where i = 0,1, 2,...
h0 =(Kavg xor k0)+r

Apply key's string length compression, where the each character of the bluff key is the result of when the sum of the first and the second ASCII characters XORed with the first string value of hi.

Mathematically,

Bi = {hi + h(i+1)} xor hi

Apply condition: If Bi < 33, Bi = Bi + 30
If Bi > 126, Bi = Bi - 30

The Keystream Generation

After the secret key is converted into a bluff key, it will then be used to generate the keystream. Generating a keystream is something has its similarity when generating the public and private key of an RSA algorithm. Instead of generating a key pair, the algorithm generates a single keystream to be used for encryption as Ill as for decryption that resembles like a symmetric cipher. Below is the keystream generation pseudo code.

First, obtain prime numbers p and q, compute the average of bluff key's array values that I denote as Biu and Lbi are for the bluff key's string length

Using Fermat and Mersenne primes respectively to obtain unique primes,

p = 2^n + 1 = 2^2 + 1 = 5
q = 2^m - 1 = 2^5 - 1 = 31

Then using RSA key generation principle obtain x = pq,

x = pq = 5(31) = 155

Then compute for totient phi(x) = (p-1)(q-1)

phi(x) = (p-1)(q-1) = (5-1)(155-1) = 4(154) = 616

Then generate the keystream using the formula below,

Ki = (phi(x) mod Bi) +30
K0 = (phi(x) mod B0) +30 = (616 mod 55) +30 = 41

Keystream (Ki) = 41 35 34 76 46

Plaintext Encryption

After I obtain the keystream value, encryption of the message takes place. For encryption and decryption algorithm, the Solitaire encryption algorithm is used with slight modification.

The first step is to convert plaintext to its equivalent ASCII code in decimal
mi: 72 101 108 108 111 32 116 104 101 114 101 33

Then subtract 32 from plaintext and keystream to ensure that values fall within the range of available ASCII code for characters,
mi = mi-32

Using Solitaire encryption technique to encrypt the plaintext, add the plaintext message to keystream numbers, modulo 94. If the sum of (mi + ki)>94 subtract 94 from the result. A value of 94 is replaced by its original value of 26 to cover the entire set of ASCII characters from (! Exclamation point) to (~ Tilde symbol).

Note: I distribute the keystream through the plain text code. If the keystream length is less than the plain text (which is most likely to occur) key will repeat its cycle

ci = (mi + Ki) mod 94

The addition of 40 to the encrypted string values ensures that the converted characters will fall on the valid ASCII characters.

Ciphertext Decryption

For ciphertext decryption, I use the session key to deciphering the encrypted message,

c = y28b=QA4[J8L, which is the ciphertext and Ki = 41 35 34 76 46 for the keystream after obtaining it from the session key. Then, convert ciphertext to its equivalent ASCII code in decimal

Next, subtract 40 & 30 from ci & Ki respectively.
Retrieving the original value ensures that the plaintext will be retrieved again during encryption.

ci = ci – 40 à ci = 81 10 16 58 21 41 25 12 51 34 16 36
Ki = Ki – 32à Ki = 41 35 34 76 46

Using solitaire decryption technique, subtract the keystream numbers from the ciphertext numbers modulo 94. If the ciphertext is less than or equal to the keystream numbers, add 94.

mi = (ci - Ki) mod 94

Then add 32 to the message's numbers to ensure that values fall within the range of available ASCII coded characters, mathematically the formula is,

$$mi = mi + 32$$

The developed hybrid encryption algorithm will then be implemented on an open source Webmail client using PHP scripts for its development platform in which there is no PGP support feature. For this study, a Roundcube mail client is the used as a test Webmail client. Then utilize GMAIL's IMAP/SMTP service for sending and receiving email messages from and to another email services online. Also, RoundCube Webmail I'll be implemented on a local machine using WAMP server as its localhost for the duration of the development.

Hybrid Encryption Algorithm Implementation on an Open Source Webmail Client

For the implementation phase of the developed hybrid encryption, a local server is installed using WAMP server for the Webmail client throughout the development of the system. Roundcube Webmail, a non-PGP open source mail client, is utilized. Configured with GMAIL's IMAP/SMTP access. The goal is to utilize the use of GMAIL's IMAP/SMTP service to enable the local Webmail to send and receive email messages to and from other existing email services over the Internet.

For this study, the aim is to integrate the developed hybrid encryption algorithm to the Webmail client. So the following strategy takes place. First, to be able to send an encrypted email message, the encryption algorithm is inserted prior of sending the message.

Next, the used of asymmetric key management protocol that enables the transfer of the session key to the recipient securely. The recipient decrypts the ciphered messages together with its formatting.

The key management protocol for this study utilizes the use of asymmetric encryption protocol using RSA key management. This protocol utilizes public key and private key. The public key is used to encrypt the message on the sender side and the private key, on the other hand, is used to decrypt the message on the recipient side.

To provide more perspective on the protocol, so this is how it works. If user A wanted to send an encrypted message to user B, user A requests the public key of user B or if user B published his pubic key. The public key then will be downloaded where user A can be able to acquire it. An alternative way of acquiring the public key is thru email. User A then adds user B's email details on address book archives together with the public key. When user A wanted to send an encrypted message to user B, the system would generate a session key as well as generating a bluff key out from the generated session key prior of encrypting the message. User B's public key will be used to encrypt the bluff key while the unencrypted bluff key will be used to generate the keystream and encrypt the message. Then user A sends the encrypted message while the system creates a temporary file in its directory that contains the encrypted message and the encrypted bluff key. While sending an encrypted message, a temporary file is created within the system's root did for this reason victory.

With this strategy, the encrypted bluff key is secured since there is no transmission of the data over the internet. As user B receives the encrypted

message, a link to the created temporary file will be provided. User B decrypts the message, as well as the encrypted message, is displayed, this is also to ensure that existing Webmail services online will be able to receive and decrypt the message. So, for this reason, the decryption is done on the sender's server. User B's private key will be used to decrypt the bluff key, and then the decrypted bluff key will then be used to decrypt the encrypted message.

Since RSA encryption system is quite slow during encryption of a lengthy message, the best strategy is to use RSA. To Encrypt the bluff key to generating the keystream on the recipient side to decrypt the message where the message is encrypted using the developed hybrid encryption algorithm using the same bluff key.

Before a user can send an encrypted message, the sender should secure the public key of the recipient first and add it to the Webmail's address book. A slight modification of the RCmail's system where the address book is now asking the recipient's public address. Composing a new message, the recipient's address box will contain the public key and the recipient's email address. The pre-encryption process takes place; the public key then extracted from the recipient's email address.

During the time of sending the message, only the encrypted message will be sent to the recipient containing the link to the temporary file stored on the mail server. So, for this reason, there is no transmission of encrypted bluff key took place that makes it more secure. When the message arrives at the recipient's end and clicks the link, the recipient will be directed to the page created as the temporary file of the mail server. The recipient will then be asking to enter his private key to be able to decrypt and read the message. While if the sent message is un-encrypted, the message will be displayed directly in the text area without any process or requirements needed. Figure 13 shows the activity diagram for decrypting the ciphertext prior of entering the private key.

RESULTS AND DISCUSSION

Security

The developed system is designed to enhance the security of two existing encryption algorithm by introducing the Bluff key generation where it hides the session key used to generate the keystream. Upon the implementation of the Hybrid encryption algorithm on an email service system using an open source Webmail client, security is further enhanced by, not sending the entire

confidential information over the internet. The recipient will only receive the encrypted message along the internet together with the link to the temporary file generated on the sender's mail server. The bluff key is secured on the server and wait for the recipients' to fetch it through the link displayed in the recipient's message area using Hypertext transport protocol to access through it.

Using this scheme ensures that a confidential message will be immune and protected from any malicious attacks on the Internet.

Speed

Hybrid Encryption Algorithm versus the RSA Encryption Algorithm

The algorithm will be tested on a localhost using mainly the algorithm itself without further encryption of the session key. A separate code is designed to test the execution time of the algorithm. Implementing the speed test on the local server ensures that there is no effect of transmission delays. For a reason that the speed of decryption will be an influenced by some technical delays within the network. Also, another aspect that can affect the time of execution is the computer hardware specifications.

Alongside time execution test, a comparison between existing encryption algorithms will also be conducted to measure and prove that the developed hybrid encryption is better than the existing one. In this test, the original RSA encryption algorithm will be used and implement it on the localhost and conduct a series of time execution test with the same input plaintext with the developed algorithm and compare its results.

For the test, different input strings, as well as formatted and non-formatted text will be used. An embedded PHP script is applied to measure the execution time with 0.000000000000001 seconds accuracy.

Execution Test between Hybrid Encryption and RSA

During the test of the Hybrid encryption algorithm alone, it has been found out that it provides as with a considerably fast execution time. But still a comparison of the existing encryption algorithm is needed to prove that the developed algorithm has any significance. On this test, both the Hybrid and the RSA encryption algorithms will be conducted on the localhost of the same machine and hardware specification as Ill as its input plaintext to ensure a fair test for both algorithms.

The test lead three different categories. First, plain unformatted text with a minimum of 800 characters, and then, a formatted text with a minimum of 800 characters and a formatted text with a maximum of 2000 characters.

The developed algorithm executes faster compared to the existing RSA algorithm at an average time difference of 0.11 seconds.

Implementation of Roundcube Webmail Client

In this test, I measured the time of execution of Roundcube Webmail client using PHP script. Although there are factors that can affect, such as the internet connection bit rates, as well as the local network where the local machine is connected. At least I grasped an idea of how fast or sluggish the system is. The preconditioned internet bandwidth of 2Mbps shows that the respond time during the test of the algorithm is 198 milliseconds considering the following hardware specifications:

Operating system:	*Windows Vista Business*
System type:	*32-bit Operating System*
Memory (RAM):	*4.00GB*
Processor:	*Pentium(R) Dual-Core CPU T4200 @ 2.00 GHz 2.00GHz*
Server type:	*WAMP on localhost*

Reliability

Reliability means how consistent the tests are and be able to acquire the same result repeatedly. For this test, three (3) different RSA key pairs, three (3) different string lengths for the message, and the type of message will be used for encryption and decryption processes. Data shows that the algorithm is reliable up to longer string lengths either unformatted or formatted messages. For string lengths greater than 1000 formatted message, in which I am using 5,500 characters it shows successful decryption. While the algorithm is pushed to its limits encrypting formatted text of about more than 5,500 characters it shows abnormalities of the decrypted message while the time of execution grows larger of about 1.03 seconds.

CONCLUSION

In this study, the researcher created a new breed of Hybrid encryption algorithm. The researcher implemented it on a non-PGP Webmail client that leads to the development of a new mode of the Webmail's capability of having a PGP feature for the first time. With faster execution time, security and reliability features, future secured systems will benefit this new breed of an encryption algorithm.

ACKNOWLEDGMENTS

This study will not be successful without the aid of some important individuals that supported and shared their unselfish expertise for the success of this study. To Dr. Victor Gregg Gabison, Associate Professor of the College of Information, Computer and Communications Technology of the University of San Jose Recoletos. The researcher is grateful to the people on the World Wide Web. To Mr. Scott also known as "skaters" for supporting this project and to the author of Roundcubemail Mr. Thomas Bruederli for responding my queries about the Webmail's PGP feature.

LITERATURE CITED

Bolhuis, M. (2014). *Using an NFC-equipped mobile phone as a token in physical access control.* Retrieved from http://essay.utwente.nl/65419/1/thesis_nfc_martijn_bolhuis_final.pdf

Gaboitsiwe, T. (2013). *Information and Communication Technology: Introduction to the Internet Components-World Wide Web and Email*. Thato Gaboitsiwe.

Jivsov, A. (2005). *Non-repudiation of e-mail messages - Networks Associates* (n.d.). Retrieved from: http://www.freepatentsonline.com/6904521.html

Lamprecht, C. J. (2012). *Adaptive security.* Retrieved from: https://theses.ncl.ac.uk/dspace/handle/10443/1435

Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of applied cryptography*. CRC press.

Park, J. S., & Sandhu, R. (2000, December). Binding identities and attributes using digitally signed certificates. In *Computer Security Applications, 2000. ACSAC'00. 16th Annual Conference* (pp. 120-127). IEEE.

Ramaraj, E. and Karthikeyan S. (2006). *A New Type of Network Security Protocol Using Hybrid Encryption in Virtual Private Networking*. J. Comput. Sci., 2: 672-675. Retrieved from http://www.thescipub.com/abstract/?doi=jcssp.2006.672.675

Ramaraj, E., et al. (2006). *An Analysis of Key Management in Cryptographic Algorithms.* Asian Journal of Information Technology 5(9): 963-967. Medwell Online, 2006. Retrieved from http:// medwellpublishing.com/abstract/?doi=ajit.2006.963.967.

Rodríguez-Henríquez, F., Saqib, N. A., Perez, A. D., & Koc, C. K. (2007). *Cryptographic algorithms on reconfigurable hardware*. Springer Science & Business Media.

Schneier, B. (2007). *Applied cryptography: protocols, algorithms, and source code in C*. john wiley & sons.

Singh, S. (2011). *The code book: the science of secrecy from ancient Egypt to quantum cryptography*. Anchor.

Tamboli, A., et al. (2011). *Simulation of Self-Organized Public-Key Management for Ad Hoc Networks* Retrieved from http://research.ijcaonline.org/nsc/number2/SPE021T.pdf

Tounsi, W., Justus, B., Boulahia, N. C., Cuppen, F., & Alfaro, J. G. (2014, June). *Probabilistic Cycle Detection for Schneier's Solitaire Keystream Algorithm. In Software Security and Reliability-Companion (SERE-C), 2014 IEEE Eighth International Conference on* (pp. 113-120). IEEE

William, S., & Stallings, W. (2006). *Cryptography and Network Security, 4/E*. Pearson Education India