Journal of Theoretical and Applied Information Technology
31st December 2013. Vol. 58 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

COST EFFECTIVE POLY VERNAM CIPHER WITH CACHE OPTIMIZATION

[1]SUNDRAM PRABHADEVI, [2]RAHUL DE, [2]PRATIK SHAH

[1] Associate Professor, Department of Computer Science and Engineering, Nandha Engineering College, India

[2] School of Computer Science and Engineering , Vellore Institute of Technology, India

E-mail: [1]s.prabhadevi@gmail.com, [2]rahul080327@gmail.com

ABSTRACT

Our digital world provides a means to access mammoths of data and services. Such a huge responsibility and task does not come without disadvantages, the most important of which is security and consistency of data and the maintenance of privacy. The science of cryptology has provided us with many means to minimize this disadvantage. This paper attempts to provide another such means to minimize that disadvantage. In this article we present a symmetric key algorithm targeting to improve the problems related to key stream generation. The security of the keystream generation of the proposed algorithm is based on Rijndael forward s – box which is manifest of closure property. The key is a set of matrices along with a tuple of coordinates, noted during the process of encryption. Cache optimization technique used eliminates the disk I/O and ensures maximum memory utilization. The performance of cache, hence the algorithm is dependent on the processor used. This algorithm is especially applicable to networks where fast and secure encryption is and decryption of data is critical.

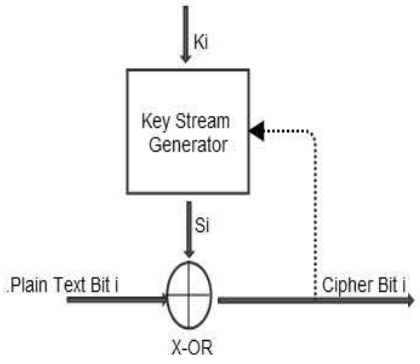
Keywords: *Poly Alphabetic, Vernam Cipher, Key Generation, Symmetric Cryptography, Cache Optimization*

1. INTRODUCTION

The importance of encryption became critical after telegraph, especially radio telegraph, was invented. Long distance communication allows information being intercepted much easier than ever. To protect the confidentiality of information, encryption is widely used in military, intelligence and diplomatic services.

Generally speaking, symmetric cryptosystems are divided into two types: block ciphers and stream ciphers. A block cipher breaks the plaintext M = (m1, m2, …, m_l) into a number of message blocks with the same length and transforms them to the cipher text C = (c1, c2, …, c_n) via an encryption function controlled by a secret key k. Stream ciphers encrypt bits individually. This is achieved by adding a bit from a *key stream* to a plaintext bit. There are synchronous stream ciphers where the key stream depends only on the key and asynchronous ones where the key stream also depends on the cipher text. If the dotted line in Fig. 1 is present, the stream cipher is an asynchronous one.

Stream ciphers tend to be small and fast, they are particularly relevant for applications with little computational resources, e.g., for cell phones or other small embedded devices. A prominent example for a stream cipher is the A5/1 cipher, which is part of the GSM mobile phone standard and is used for voice encryption.



Stream ciphers may be faster or have a smaller implementation footprint than comparable block ciphers. They operate more naturally on data of short, variable or unknown length. Finally, the keystream generation is completely independent of the plaintext data, and so it may be computed in parallel with or in advance of the data stream. In general, it is also useful for system designers to have a reasonable selection of different encryption

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

algorithms to choose between, as this makes it possible both to select precise performance tradeoffs suitable for a specific application.

2. RELATED WORK

2.1 One-time Pad and Stream Ciphers

The one-time pad, also called Vernam's cipher (Vernam, 1926), was invented by Vernam in 1917. The bit-wise one-time pad is easy to illustrate. A one-time key is randomly generated and it is as long as the message. The key is XOR-ed with the plaintext for encryption, and the key is XOR-ed with the cipher text for decryption.

The one-time pad is the only encryption algorithm that is unconditionally secure. The perfect secrecy of one-time pad was proved by Shannon (1949). Although the one-time pad is perfectly secure, it is inconvenient to use in many applications due to the constraints that the key is too long and each key can be used only once. A strong synchronous stream cipher is a good replacement for the one-time pad. A stream cipher can be used to generate many key streams from the same key and different initialization vectors, and then each keystream can be used to encrypt a message.

Stream Ciphers

RC4, also known as ARC4 or ARCFOUR is a stream cipher developed by Ron Rivest in 1987. The most widely used stream cipher around is, by far, RC4, is extremely fast in software and can be implemented in just a few lines of code. This cipher is used to provide security in the popular protocols such as the TLS (Transport Layer Security) and WEP. This cipher is highly simplistic and uses two major phases. The first is the key scheduling algorithm and the second is the pseudo random key generation. The two phases are used to generate the encrypted stream of bits which is then used to encrypt the incoming stream of input.

There are many such stream ciphers as the HC-128 (Wu, 2008), supporting 128-bit keys, Rabbit (Boesgaard *et al*., 2008), supporting 128-bit keys, Salsa20/12 (Bernstein, 2008), supporting 128 and 256-bit keys, SOSEMANUK (Berbain et al., 2008), supporting 128-256 bit keys, Trivium (Canniere and Preneel, 2008),supporting 80-bit keys, Grain v1 Hell et al., 2008), supporting 80-bit keys, MICKEY v2 (Babbage and Dodd, 2008), supporting 80-bit keys. RC4 but the main problem faced by all of the existing stream ciphers today is the fact that the key stream generated is not unique enough when the cipher tries to match the input data stream with the key stream.

RC4 specially has a key generation weakness, which is exploitable as is evident from (Daemen and Rijmen, 2002). RC4 being a part of the standard SSL encryption standard must be as safe as possible. Also the fact that all the stream ciphers are somewhat based on the Fiestel networks which though makes it faster but at the same time reduces their randomness and makes the entire process kind of predictable. Such a weakness is not an option in highly secure networks.

3. PROPOSED WORK

The proposed work, Poly Vernam cipher greatly increases the randomness by employing a variety of methods such as the Vernam ciphers, Poly alphabetic Substitutions (Alberti, 1997), Substitution-Permutation Networks (Melville, 2012) and the well-known Rijndael S-Boxes to permute the key. The key itself is securely generated and made as random as possible. We used the Rjindael S-box (Daemen and Rijmen, 2002), which mathematically guarantees a closure property, and the fact that even after the transformation proposed by us the closure property is still maintained. Cryptanalysis of Rjindael S-box in an overdefined system of algebraic equations with probability 1 (Courtois, N. and J. Pieprzyk, 2002) is practically impossible in high speed networks. Statistically speaking a stream cipher is that not only optimizes memory and cache usage but also produces key stream, which is equal to the data stream thus ensuring a character-to-character mapping which is random in nature. The algorithm targets the nodes of the network it is operating on and the fact that there can be nodes of varied architecture and processing powers, which are inter-linked. This algorithm takes into account the amount of processing power and memory available at every node thereby ensuring seamless operation, which is both compatible and efficient enough. We also have proposed a variable buffer size as an input to the cipher, which is non-existing in the current system. This is the key factor, which enables the algorithm to adapt to the under-lying hardware. Real life networks consist of stand-alone nodes, which not necessarily have the necessary resources to run a generalized network stream cipher because of the fact that all the current ciphers and socket level security system use a fixed resource-consuming algorithm and depend on various pre-requisites, which might not be available, and hence our solution is better than ones before. Our algorithm uses custom Huffman based encoding to reduce the output size and actually

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

takes into account that the real life data can be repetitive in nature and tried to identify such repetitions and reduce the load on the network. Decryption is straightforward and the key and data can be sent via different paths.

4. DESIGN OF POLY VERNAM CIPHER

The algorithm works on the individual byte level that is operates on 8 bits of data (0 - 255) range. The algorithm operates in the following rounds:

Key Generation Round

- A 256 * 256 matrix is generated where each column has numbers from 0-255 indicating the 256 ASCII characters.
- Each of the columns is randomly shuffled producing a matrix of permuted columns each having values from zero.

Encryption

- A randomly chosen point in the matrix say (x,y). Each point has eight neighboring cells hence 8 directions for a point to move in. We substitute these directions with numbers from 0-7 refer Table 1. The starting coordinates are saved as the direction.

Table 1: Direction Substitution Table

Number(Bits)	Direction
0	Top
1	Top Right
2	Right
3	Bottom Right
4	Bottom
5	Bottom Left
6	Left
7	Top Left

- The input stream is read character by character and the position selected is moved along the direction chosen in the matrix. Each character encountered in the traversal of the matrix is XOR-ed with the character in the input stream and the position is incremented in the direction.
- At a particular iteration depending on the length of the input stream, the read position of the matrix reaches the boundaries of the current matrix. The next generation key is now generated.
- The next generation of the key is generated by utilizing the Rjindael forward S-Box (as in Table 2) transformation on the current matrix. Given the properties of the transform, a new matrix is obtained which is revertible back to the previous version. This is the next level matrix, which is used as a crypto matrix for the individual characters from the input stream.
- The starting point of this matrix is the same point as the place where the last read position collided with the boundaries of the last matrix. A new direction is chosen from the degrees of freedom available and the similar method is followed as mentioned above.
- The same method is followed till another collision happens with the current matrix's boundaries and the aforementioned steps are repeated until the entire message is encrypted.
- The key generated for this particular encryption session is the initial 256*256 permuted matrix, the initial starting point of the matrix then followed a tuple having the directions the encoder took while encrypting the data.

Decryption

- The key is read and the 256 *256 matrix is reconstructed followed by the starting position and the direction. The read pointer is placed on that position and read moved along the matrix in that direction XOR-ing the current input character with the current matrix character.
- Just like the encryption round, the decryption works by following the matrix data along the particular direction until a collision takes place at the boundaries and the subsequent generation of matrices are produced which decrypts the data.

4.1. Alternative Conservative Method with Redundancy Reduction Round

The above suggested algorithm produces a fair amount of overhead in terms of the multiple level matrix generation and when smaller and limited resource execution environments are taken into consideration, this is a problem. To overcome this, a single matrix repetitive method is also proposed. This method reduces both memory and processor footprint at the slight cost of reduction in randomness in case

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

of redundant inputs. The following steps are involved for the alternative method:

Table 1: Rijndael Forward S - Box

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Key Generation Round

- The initial matrix here is exactly same as the aforementioned method.
- A reverse matrix is also generated which contains the opposite characters in the same places as the first matrix. For example, if in the above matrix in the first column, at the 97th position, 255 is there, then in the inverse matrix will have 97 in the 255th position of the first column and so on.
- The Key consists of the inverse matrix and the starting point, which is essentially a randomly chosen column.

Encryption

- Randomly a column is chosen and is selected as the starting point and is noted.
- The input stream is read character by character and each of them is replaced with the corresponding character from the matrix. For example, if 'a' is encountered in the input stream and the starting column in the matrix is the 10th one, the 'a' will be replaced with the character at the 97th position of the 10th column.
- The read position is then incremented to the next column and reading a character from it and incrementing the column further processes the input stream.
- When the column count reaches 255 i.e. the end of the matrix, it is simply reset back to zero and it continues producing a stream of encrypted characters.

Decryption

- The inverse matrix is read from the key and the read head is positioned on the column number, which too was presented in the key.
- The encrypted stream is read character by character and the same method of replacing the input character with the corresponding character in the inverse matrix to get a stream of decrypted character.

Redundancy Reduction Round

The shortcoming with this method is that if the input is highly redundant this method tends to produce repeated blocks of cipher text, which results in a relative easy guess of the key itself. To address this issue the Redundancy Reduction Round is used which is as follows:

- The entire input stream is buffered into a pre-defined block.
- The well-known Huffman Codes is used to reduce the input redundancies and try to compress the input characters into blocks of non-repeating characters.
- The blocks of non-repeating characters are used as an input stream to the aforementioned algorithm, which operates on it producing a reduced set of encrypted characters, which exactly corresponds to the possibly longer input stream.
- Huffman decoding follows the decoding stage to get the original expanded output.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.2 Unique Key Generation and the Vernam Ciphering

The randomness of the algorithm depends on the fact that for each successful decryption the key used is to be destroyed, never to be used again. A central database of the unique ids of the keys is to be maintained which keeps a track of the keys generated. The method used to achieve this is via the SHA 256 fingerprinting.

- The initial 256 X 256 matrix is linearized into a (256 ^ 2) length string and is used as the input for the SHA 256.
- A fixed length 256-bit hash is obtained which is then inserted into the key database provided it's unique. Else another initial key is generated which is again verified through same process until a unique SHA 256-bit fingerprint is obtained.
- Such a method is highly reliable and secure in a centralized network system.
-

5. CACHE OPTIMIZATION

We have proposed and implemented a method to optimize cache utilization depending on the cache size of machine on which the code is run.
To improve the I/O speeds for the algorithm, first the underlying CPU is queried for the available levels of cache. The largest cache amongst is chosen as the size of the I/O buffer size. For example a CPU has three levels of cache and L1 is 32K, L2 being 64Kand L3 is 3M. Therefore, the I/O buffer size for out algorithm will be 3 MB. The advantage of using this size is to fill up the entire cache with the data with a single read. As the read values will be bytes ranging from 0-255, the probability of all the 256 bytes being in the cache is high. The is very crucial for speed as the consecutive reads of 3 MBs will result I the request of the same bytes in a different order and since they are all there in the cache, disk I/O is eliminated hence greatly speeding up the process and also making proper use of all the resources at the same time.

6. SECURITY ANALYSIS

As the Poly Vernam Cipher uses a random key shuffling mechanism including a random direction mechanism a simple brute force attack on this cipher would almost be impractical.
The shuffling of the key itself results in 256!^256 possibilities of the key. Furthermore the random selection of the start point has ((254*254*8) + (4*254*5) + (4*3)) possibilities and at every collision there are a minimum of 3 and a maximum of 5 possibilities so in the worst case for n collisions there are n*5 possibilities.
So in total for a worst-case brute force scenario it would require ((256!^256) * ((254*254*8) + (4*254*5) + (4*3)) *5*n) hits to key the key right. Here n is number of collisions which is dynamic.
Hence Poly Vernam cipher can be certified as immune to brute force as this cipher is mainly targeted for the networks and high speed networks like vehicular ad hoc networks and in such scenarios a brute force would take too long to crack the cipher and thus would render the cracked key to be useless.

7. CONCLUSION

Most people would argue that the study of cryptography has reached a pinnacle and is now saturated and they would be completely correct. With the introduction of the AES cipher we have reached the maximum level of security that ciphers can provide us on traditional computing systems. This cipher is not meant to be comparable to AES ciphers but it is meant to be an alternative in situations where AES takes too long a time to be implemented. In such cases the unique features of this cipher such as Cache Optimization and redundancy check would provide a tremendous speed boost to the encryption and the decryption process and is therefore ideal in situation where speed is of essence in networks.

REFRENCES:

[1] L.B. Alberti, "A Treatise on Ciphers. Trans. A. Zaccagnini" Foreword by David Kahn, Galimberti, Torino, 1997.

[2] S. Babbage, and M. Dodd, "The MICKEY Stream Ciphers", Lecture Notes Computer Science, 4986: 191-209, http://www.ecrypt.eu.org/stream/mickeypf.html, 2008

[3] C.Berbain, O. Billet, A. Canteaut, N. Courtois and H. Gilbert, "SOSEMANUK, A Fast Software-Oriented Stream Cipher", Lecture Notes Computer Science, 4986: 98-18. http://www.ecrypt.eu.org/stream/sosemanukpf.html, 2008

[4] D.Bernstein, " The Salsa20 Family of Stream Ciphers", Lecture Notes Computer Science, 4986: 84-97. http://www.ecrypt.eu.org/stream/salsa20pf.html, 2008

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[5] M.Boesgaard, M. Vesterager and E. Zenner, " The Rabbit Stream Cipher", Lecture Notes Computer Science, 4986: 69-83. http://www.ecrypt.eu.org/stream/rabbitpf.html, 2008

[6] Canniere and B. Preneel, "TRIVIUM", Lecture Notes Computer Science, 4986: 244-266. http://www.ecrypt.eu.org/stream/triviumpf.html, 2008

[7] N.Courtois and J. Pieprzyk, "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations", pp: 267-287, 2002

[8] J. Daemen and V. Rijmen, " The Design of Rijndael: AES-The Advanced Encryption Standard", Springer, ISBN-10: 3-540-42580-2, 2002

[9] M.Hell, T. Johansson, A. Maximov and W. Meier, "The Grain Family of Stream Ciphers", Lecture Notes Computer Science 4986: 179-190. http://www.ecrypt.eu.org/stream/grainpf.html, 2008

[10] K.Melville, "Securing Record Communications: The TSEC/KW-26", 2012

[11] C. Shannon, "Communication Theory of Secrecy Systems. Bell System Technical Journal", 28: 656-715, 1949

[12] G.S.Vernam, "Cipher Printing Telegraph Systems For Secret Wire And Radio Telegraphic Communications", IEEE Journal, 55: 109-114, 1926

[13] H.Wu, "The Stream Cipher HC-128", Lecture Notes Computer Science, 4986: 39-47, http://www.ecrypt.eu.org/stream/hcpf.html, 2008