Proceedings of the 4th National Conference; INDIACom-2010
Computing For Nation Development, February 25 – 26, 2010
Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi

Comparative Analysis of Structures And Attacks on Various Stream Ciphers

Mohammad Ubaidullah Bokhari
Chairman, Department of Computer Science AMU Aligarh (India), E-Mail: mubokhari@gmail.com
Faheem Masoodi
Research Scholar, Dept. of Computer Science AMU Aligarh (India), E-Mail: masoodifahim@gmail.com

ABSTRCAT
In the recent past, research work has been done in the area of steam ciphers and as a result of which, many design models for stream ciphers were proposed. In Order to realize a world standard for data encryption that would prove good in the due course of time, withstand the action of cryptanalysis algorithms and strengthen the security that will take longer to be broken, we took up this research work. In contrast to block ciphers, stream ciphers do not have a standard model and a variety of structures are followed in their design. In this review we try to examine the different design philosophies adopted by various stream ciphers, various attacks carried out on these stream ciphers and the effect of these attacks. This work also gives an insight into the recent trends in the design of stream ciphers.

1.0 INTRODUCTION

Does increased security provide comfort to paranoid people? Or does security provide some very basic protections that we are naive to believe that we don't need? Today when tens of millions of people rely on Internet for essential communication and trade & commerce between them, a secure system becomes a very important issue to deal with. Cryptography under such circumstances forms an essential aspect for secure communications. Cryptography deals with four major goals viz Confidentiality, Data integrity, Authentication and Non-repudiation and thus is widely used to secure telephonic messages, e-mails, credit card information, and corporate data[1] but with all these applications under its sleeve, one must keep in mind that cryptography on its own does not suffice all the requirements of security. Cryptography systems can be broadly classified into symmetric-key systems (AES,RC4,DES) that use a single key that both the sender and recipient have, and public-key or asymmetric systems (ElGamal, McEliece, RSA) that use two keys, a public key known to everyone and a private key that only the recipient of messages uses[3]. Symmetric cryptosystems is usually divided into block ciphers and stream ciphers. Rueppel [2] describes the difference as:

"Block ciphers operate with a fixed transformation on large blocks of plain-text data; stream ciphers operate with a time-varying transformation on individual plain-text digits".

This classification is not absolute, and any block cipher can be used as a stream cipher by using certain modes of operation. A stream cipher is first loaded with an s-bit initial state S_0 and a kc-bit cipher key K_c. At each time t, the output function O of the stream cipher generates a key stream digit Z_t. This key stream digit is used to encrypt a plaintext digit P_t with an encryption function E and we obtain the ciphertext C_t. Then the state S_t is updated by the state updates function U. The encryption process is thus governed by the following three equations:

$$Z_t = O(S_t, K_c)$$
$$C_t = E(P_t, Z_t)$$
$$S_{t+1} = U(P, S_t, K_c),$$

Where the encryption function E is such that it is easy to construct a decryption function D, the decryption process can be described as follows:

$$Z_t = O(S_t, K_c)$$
$$P_t = D(C_t, Z_t)$$
$$S_{t+1} = U(P_t, S_t, Kc) .$$

As stated in [4], there are several benefits of stream ciphers compared to block ciphers:

- Stream ciphers are generally much faster than block ciphers
- No or limited error propagation
- Low hardware complexity
- The keystream can be sometimes generated prior to encryption/decryption.(in the synchronous case)

Further on, Stream ciphers can be classified based on internal state as being either synchronous or self synchronizing. If the change in state occurs independent of the plaintext or cipher text messages the cipher is categorized as a synchronous stream cipher. In contrast, self-synchronizing stream ciphers update their state based on previous cipher text digits. In case of synchronous ciphers, the keystream generated is dependent only on the key and the position i while as in case of self-synchronous the keystream depends only on the key and a fixed amount of previous ciphertext. Synchronous ciphers are described as having no error propagation while error propagation is limited in self-synchronous With synchronous ciphers, synchronization is achieved with 'marker positions' in the transmission, in contrast self–synchronizing ciphers have the facility to resume correct decryption if the keystream falls out of synchronization. Though desirable properties are found in both the variations, various implications are found in both of these. During decryption, the synchronous cipher limits the opportunity of detecting an error and a more serious limitation is that the attacker is able to make controlled changes to parts of ciphertext knowing very well the effect being induced on the corresponding plaintext. Rueppel [5] suggests two implications with self-synchronizing ciphers. One is that, an opponent is aware of some of the variables being used as input to the generator, as it is taken from the ciphertext. Another is

 ISSN 0973-7529 ISBN 978-81-904526-9-4

that these generators have a limited analyzability because the keystream depends on the message.

2.0 DIFFERENT DESIGN APPROACHES TO STREAM CIPHERS

In the absence of some standard model, a great variety of structures can be found in literature. Some of these structures are:

2.1 Linear Feedback Shift Register based stream ciphers

Linear feedback shift registers (LFSRs) form the quite common components in stream ciphers as they can be implemented cheaply in hardware. They possess simple structure that is easy to analyze mathematically and their properties are well-understood. The basic motivation behind the frequent usage of LFSRs in stream cipher design is their simple structure. However, LFSRs do not guarantee adept security. The various general schemes like filtering function, clock controlling and non-linear combining function can be used in order to increase the security of LFSRs.

2.2 Non-linear combining functions

Feedback registers with non-linear functions are becoming an important component of some modern stream ciphers. LFSRs are inherently linear. This linearity can be removed by feeding the outputs of several parallel LFSRs into a non-linear Boolean function to form a *combination generator*. The various properties of such a *combining function* are critical for ensuring the security of the resultant scheme, like avoiding correlation attacks.

2.3 Clock-controlled generators

Clock controlled generators serve the function of introducing non-linearity in LFSRs. This non-linearity is achieved by having LFSR clocked irregularly by being driven by the output of some other LFSR. Several generators based on this principle have been proposed like stop-and-go, alternating step generator and the shrinking generator.

2.4 Filter generator

Filtering Boolean functions is yet another approach of removing the linearity of LFSR and thus improving its security. Although, not sufficient to be resistant enough against several attacks, certain characteristics are supposed to be necessary in stream ciphers with this structure. These characteristics include: high algebraic degree, high non-linearity, balance, and algebraic immunity.

2.5 T-Function Usage

Klimov and Shamir in 2002 proposed a new class of invertible mapping termed as triangular functions(T-functions) with a single cycle on n bit words. Though these single word t-functions are not very useful because of their bit size, however their expanded version of multi-word T-function is proposed and is already used in a number of stream ciphers as a replacement for LFSR.

3.0 ANALYSIS OF SOME STREAM CIPHERS

3.1 RC4

Ron Rivest in 1987 designed, what is probably the most widely used stream cipher, known as RC4. It was actually designed with the intent of providing security to RSA and as a result was long held as a trade secret but in 1994 source code was anonymously leaked to the cipher punks mailing list.RC4 is used in the SSL/TLS standard for secure communication between web browsers and servers and in the WEP protocol used in 802.11 wireless LAN

3.1.1 Implementation

RC4 is ideal for software implementation, as it requires only byte manipulations. The inner state consists of a dynamically changing table S and two byte variables. It uses 256 bytes of memory for the state array, S[0] through S[255], k bytes of memory for the key, key[0] through key[k-1].A key of variable length K is used to permute S, which initially contains the values 0….255 in an ascending order. Output is then generated depending on the state and in each step some elements in S are permuted. The huge state size of log (256!)≈1684 bit seems to rule out the linear cryptanalysis.

3.1.2 Security

Somewhat surprisingly for such a widely known and analyzed cipher, Mantin and Shamir found a trivial distinguishing attack as late as 2001[8]. The first few hundred output bytes are random and leak information about the key, especially the first few bytes are highly biased and the second output byte of RC4 takes on the value 0 with probability 2-7 instead of 2-8. The reason for these weaknesses is that the table S does not have uniform distributions after the initial permutation. Mossel et al [9] showed that for a table of size n, shuffling by semi-random transpositions has a computation complexity Θ (n log n) to get table uniformly permuted.

3.2 Polar Bear

Kohan H Astad and Mats N Aslund designed Polar Bear with the claim that it was suitable for both hardware and software. Polar bear is one of the 35 candidates submitted to eSTREAM.

3.2.1 Implementation

Polar Bear uses one 7-word LFSR R^0 and one 9-word LFSR R^1, besides these registers, the internal state of the cipher also depends on a word quantity S and a dynamic permutation of bytes D. The cipher is primarily designed for a key length of 128 bits. The IV can be any number of bytes up to a maximum of 31. On each message to be processed, the cipher is initialized by taking the key, interpreting the IV as a plaintext block, and applying a (slightly modified) five round Rijndael encryption with block length 256. The resulting cipher text block is loaded into R^0 and R^1. Finally D_8 is initialized to equal the table T_8, the Rijndael S-box and S is set to zero. Output is produced 4 bytes at a time. To this end, the two LFSRs are first irregularly clocked, determined by S. Eight bytes selected from R^0 and R^1 are run through the permutation D_8 to produce the four output bytes. Selected entries in D_8 are swapped. Finally S and R^0 are modified in preparation for the next output cycle. Entries in R^1 are not modified apart from the LFSR stepping.

3.2.2 Security

In his research paper John Mattson [10] has shown certain weakness of the stream cipher polar bear. He has written that

Continued on Page No. 236

there are several unfortunate coincidences that make these attacks possible.

Continued on Page No. 238

It is relatively straightforward to see that the attack resistance of Polar Bear does not meet its key size. For instance, by guessing one (the shorter LFSR value, it is possible to deduce the value of other by observing output. Hence, we have a complexity about 2^{112}. A solution to this would be to make one, or both of the LFSRs longer, but the LFSRs would have to be almost double the length to withstand attacks like described above.

3.3 Sober t-32

Sober t-32 is an asynchronous stream cipher design for a secret key that of 256 bits. The SOBER ciphers were primarily designed for the use of mobile telephony. In an application such as mobile telephony, any loss of synchronization with such a stream cipher is disastrous. To counter the loss of synchronization, information is sent in small portions called frames.

3.3.1 Implementation

Sober t-32 has four components: key loading, an LFSR, a nonlinear filter (NLF) and stuttering. The key loading sets the 17 words in the register of the LFSR to an initial state derived from the key. In some cases a re-synchronization key or frame key is used during key loading. The LFSR uses a linear feedback function to construct a stream of words; this stream of words is the LFSR stream. The process of producing a new word in the LFSR stream is called a cycle of the LFSR. The purpose of NLF is to disguise the linearity in the LFSR stream. After every cycle of the LFSR, the NLF combines words from the register in a non-linear function; the outputs form the NLF stream. The stuttering uses occasional NLF stream words to select NLF stream words to be used in the key stream.

3.3.2 Security

In the research paper cryptanalysis of sober-t32 [11] by Steve Babbage and his team, have shown some new attacks on the stream cipher sober-t32, these attacks include guess and determine attack on un-stuttered sober-t32 and distinguishing attack on full sober-t32. The first attack is a $2^{252.21}$ Guess and Determine attack on un-stuttered sober-t32. This attack is due a probabilistic property of the t-class of stream ciphers found in their s-box construction. This relationship between 8 Bits in and 8 bits out is not diffused to other positions in the word. Even a Cyclic shift at the end of the S-Box would have destroyed the attack. In order to prevent similar attacks, suggestion is that in word-based LFSR's, the NLF Should Implicate the Whole Word and not just A Part of the Word in Sober-t32. Then the attacker will not gain any profit by guessing some bits of the words. Stuttering prevents the attack- not so much by the Uncertainty it introduces, as by the fact that consecutive words don't appear in the Key Stream. In fact, a timing attack [12] on the Stuttering can reveal a long sequence of consecutive words that are not eliminated, thus enabling the guess and Determine Attack. Next, two ways of mounting distinguishing attacks on full Sober-t32 have been elaborated. Both attacks are based on attacks described in [13]. The first attack is an adaptation of the attack on un-stuttered Sober-t32, such that it also works on full Sober-t32. This attack could distinguish the Sober-t32 key-stream from a uniform source with about 2^{200} output words.

4.0 CONCLUSION

Undoubtedly, the importance of stream ciphers in computer applications cannot be ignored. Therefore, a standardized model for the stream cipher design is certainly today's requisite. Though some new ideas are being practiced but the classical structures like LFSR, clock controlling etc can let us make a good start in this field. This review studies the standard structures and a few important stream ciphers with a hope to come up with an advanced stream cipher that meets the standards of efficiency in terms of security, implementation, and speed and error propagation in future.

5.0 REFERENCES

[1]. "Stream Ciphers" RSA Laboratories Technical Report TR-701Version 2.0|July 25, 1995 M.J.B. Robshaw.

[2]. Rainer A. Rueppel. "Stream ciphers" The science of information integrity, IEEE press, New York 1992.

[3]. JOHN MATTSSON. "Stream Cipher Design" Master of Thesis Stockholm, Sweden 2006

[4]. Marcus Schafheutle and Stefan Pyka. Stream Ciphers. Technical report, NESSIE consortium, February 2003.

[5]. R.A. Rueppel. New Approaches to Stream Ciphers. PhD thesis, Swiss Federal Institute of Technology, Zurich, 1984

[6]. "An Overview of Cryptography" By *Gary C. Kessler* 13 December 2009

[7]. "Comparative Analysis of the Structures of eSTREAM Submitted Stream Ciphers", by Mehreen Afzal, Firdous Kausar, Ashraf Masood, IEEE-ICET 2006

[8]. Itsik Mantin and Adi Shamir, A Practical Attack on Broadcast RC4.

[9]. Elchanan Mossel, Yuval Peres, and Alisyiar Sincliar. "Shuffling by semi random transportations", 2004

[10]. Pig Bear, Stream Cipher Design-An evaluation of the eSTREAM candidate Polar Bear, John Mattson, PhD thesis 2006 Sweden.

[11]. Steve Babbage, Christophe De Canniuere, Joseph Lano, Bart Preneel, and Joos Vandewalle. Cryptanalysis of SOBER-t32. In T.Johansson, editor, Fast Software Encryption, FSE 2003, number 2887 in Lecture Notes in Computer Science, pages 111{128. Springer-Verlag, 2003.

[12]. Patrik Ekdahl and Thomas Johansson. Distinguishing attacks on SOBER-t16 and t32. In J.Daemen and V.Rijmen, editors, Fast Software Encryption, FSE 2002, number 2365 in Lecture notes in Computer Science, pages 210{224. Springer-Verlag, 2002.

[13]. Steve Babbage, Christophe De Canniuere, Joseph Lano, Bart Preneel, and Joos Vandewalle.Distinguishing attacks on SOBER-t32. In proceedings of the 3rd NESSIE Workshop, page 14, 2002.

[14]. Wikipedia, the free Encyclopedia.

 ISSN 0973-7529 ISBN 978-81-904526-9-4