Performance Analysis Of Secured Synchronous Stream Ciphers

Padmalatha Eddla[1] , R.Ravinder Reddy[2]
[1,2]Computer Science Department,
CBIT, Hyderabad, A.P, India.
E-mail : [1] padmalathae.cbit@gmail.com, [2] ravi.ramasani@gmail.com,

Abstract : ***The new information and communication technologies require adequate security. In the past decades ,we have witnessed an explosive growth of the digital storage and communication of data ,triggered by some important breakthroughs such as the Internet and the expansive growth of wireless communications. In the world of cryptography ,stream ciphers are known as primitives used to ensure privacy over communication channel and these are widely used for fast encryption of sensitive data. Lots of old stream ciphers that have been formerly used no longer be considered secure ,because of their vulnerability to newly developed cryptanalysis techniques. Many designs stream ciphers have been proposed in an effort to find a proper candidate to be chosen as world standard for data encryption.***

From these designs, the stream ciphers which are Trivium,Edon80 and Mickey are implemented in 'c' language with out affecting their security .Actually these algorithms are particularly suited for hardware oriented environments which provides considerable security and efficiency aspects. We will be targeting hardware applications, and good measure for efficiency of a stream cipher in this environment is the number of key stream bits generated per cycle per gate. For good efficiency we are approaching two ways .One approach is minimizing the number of gates.The other approach is to dramatically increase the number of bits for cycle. This allows reducing the clock frequency at the cost of an increased gate count. Apart from the implementation the analysis which includes the security of these algorithms against some attacks related to stream ciphers such as guess and deterministic attacks, correlation attacks, divide and conquer attacks and algebraic attacks are presented.

Keywords: Cryptanalysis, Trivium, Edon80, Mickey.

1. Introduction Cryptology is the science that aims to provide information security in the digitalworld. Cryptology is the science that aims to provide information security in the digital world. A thorough overview of cryptology can be found in the Handbook of Applied Cryptography by Menezes [1]. Information security comprises many aspects, the most important of which are confidentiality and authenticity. *Confidentiality* means keeping the information secret from all except those who are authorized to learn or know it. *Authenticity* involves both ensuring that data have not been modified by an unauthorized person (*data integrity*) and being able to verify who is the author of the data (*data origin authentication*).

Cryptology is usually split up into two closely related fields, cryptography and cryptanalysis. *Cryptography* studies the design of algorithms and protocols for information security. The ideal situation would be to develop algorithms which are provably secure, but this is only possible in very limited cases. Therefore, the best way to assess the security of a cryptographic algorithm or protocol is by testing all possible known attacks on it. *Cryptanalysis* is concerned with this study of mathematical techniques that attempt to break cryptographic primitives.

The cryptographic algorithms are usually split up into two families, symmetric algorithms and public-key algorithms. *Symmetric algorithms* (also called secret- key algorithms) require that the same secret key is shared by the communicating parties. In *public-key algorithms* (also called private-key algorithms), the public key *e* is made public and the private key *d* is kept secret by a single entity.

There are two classes of symmetric-key encryption schemes which are commonly distinguished: *block ciphers* and *stream ciphers*. A *block cipher* is an encryption scheme which breaks up the plaintext messages to be transmitted into strings (called *blocks*) of a fixed length t and encrypts one block at a time. In *stream ciphers*, it is same as block ciphers with a block length of a bit or a byte.

Stream ciphers form an important class of symmetric-key encryption schemes. Let K be the key space for a set of encryption transformations. A sequence of symbols $e_1e_2e_3.....e_i \in K$, is called a *key stream*.

Let A be an alphabet of q symbols and let E_e be a simple substitution cipher with block length 1 where $e \in K$. Let $m_1m_2m_3...$ be a plaintext string and let $e_1e_2e_3.....$be a key stream from K. A *stream cipher* takes the plaintext string and produces a cipher text string $c_1c_2c_3.....$ where $c_i = E_{ei}(m_i)$. If d_i denotes the inverse of e_i, then $D_{di}(c_i) = m_i$ decrypts the cipher text string. LFSR with a primitive feedback polynomial is also called a maximum- length LFSR and the sequence produced is called a maximum length sequence.

Sequences generated by maximum length LFSRs have good statistical properties, desirable for key stream generators construction, but we need to destroy the linearity, i.e. increase the linear complexity, before the sequence can be used.

2. Trivium

Trivium is a synchronous stream cipher designed to generate up to 2^{64} bits of key stream from an 80-bit secret key and an 80-bit initial value (IV). As for most stream ciphers, this process consists of

two phases: first the internal state (288 bits) of the cipher is initialized using the key and the IV, then the state is repeatedly updated and used to generate key stream bits.

2.1 Key stream generation

The proposed design contains a 288-bit internal state denoted by ($s_{1,\ldots\ldots}s_{288}$). The key stream generation consists of an iterative process which extracts the values of 15 specific state bits and uses them both to update 3 bits of the state and to compute 1 bit of key stream z_i. The state bits are then rotated and the process repeats itself until the requested N<= 2^{64} bits of key stream have been generated. Here N is the length of the stream. A complete description is given by the following simple pseudo-code:

for i = 1 to N do
$t_1 \leftarrow s_{66} + s_{93}$
$t_2 \leftarrow s_{162} + s_{177}$
$t_3 \leftarrow s_{243} + s_{288}$
$z_i \leftarrow t_1 + t_2 + t_3$
$t_1 \leftarrow t_1 + s_{91} \_ s_{92} + s_{171}$
$t_2 \leftarrow t_2 + s_{175} \_ s_{176} + s_{264}$
$t_3 \leftarrow t_3 + s_{286} \_ s_{287} + s_{69}$
$(s_1; s_2; : : : ; s_{93}) \leftarrow (t_3; s_1; : : : ; s_{92})$
$(s_{94}; s_{95}; : : : ; s_{177}) \leftarrow (t_1; s_{94}; : : : ; s_{176})$
$(s_{178}; s_{279}; : : : ; s_{288}) \leftarrow (t_2; s_{178}; : : : ; s_{287})$
end for

Note that here, and in the rest of this document, the `+' and `_' operations stand for addition and multiplication over GF(2) [11] (Galva Field) (i.e., XOR and AND), respectively. A graphical representation of the key stream generation process can be found in Fig. 2.1.

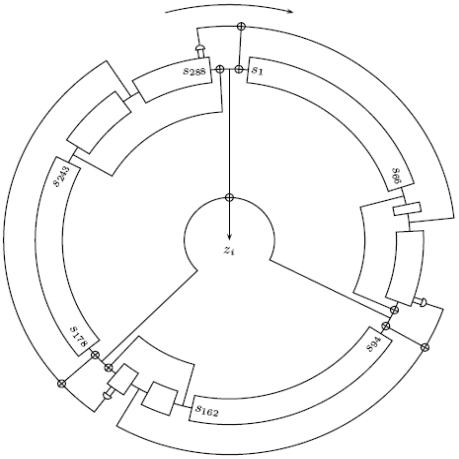

Figure 2.1 key stream generation

2.2. Key and IV setup

The algorithm is initialized by loading an 80-bit key and an 80-bit IV into the 288-bit initial state, and setting all remaining bits to 0, except for s_{286}, s_{287}, and s_{288}. Then, the state is rotated over 4 full cycles, in the same way as explained above, but without generating key stream bits. This is summarized in the pseudo-code below:
$(s_1; s_2; : : : ; s_{93}) \leftarrow (K_1; : : : ;K_{80}; 0; : : : ; 0)$
$(s_{94}; s_{95}; : : : ; s_{177}) \leftarrow (IV_1; : : : ; IV_{80}; 0; : : : ; 0)$
$(s_{178}; s_{279}; : : : ; s_{288}) \leftarrow (0; : : : ; 0; 1; 1; 1)$
for i = 1 to 4 * 288 do
$t_1 \leftarrow s_{66} + s_{91} \_ s_{92} + s_{93} + s_{171}$
$t_2 \leftarrow s_{162} + s_{175} \_ s_{176} + s_{177} + s_{264}$
$t_3 \leftarrow s_{243} + s_{286} \_ s_{287} + s_{288} + s_{69}$
$(s_1; s_2; : : : ; s_{93}) \leftarrow (t_3; s_1; : : : ; s_{92})$
$(s_{94}; s_{95}; : : : ; s_{177}) \leftarrow (t_1; s_{94}; : : : ; s_{176})$
$(s_{178}; s_{279}; : : : ; s_{288}) \leftarrow (t_2; s_{178}; : : : ; s_{287})$
end for.

2.3 Implementation Hardware

Trivium is a hardware oriented design focused on flexibility. It aims to be compact in environments with restrictions on the gate count, power -efficient on platforms with limited power resources, and fast in applications that require high-speed encryption.

Software

Despite the fact that Trivium does not target software applications, the cipher is still reasonably efficient on a standard PC.

3. Edon80

It is a synchronous stream cipher which takes the input as a key size of 80 bit and Initial Value also has 80 bit length. It is not related to any LFSR based cipher. It will produce 2 bits at a time as a key stream instead of one bit in other stream ciphers as given in Trivium. This is also targeted for hardware configurations shown in fig3.1.

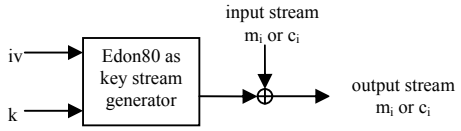

Fig3.1Graphical representation of Edon80 as binary additive stream

The internal structure of *Edon80* can be seen as pipelined architecture of 80 simple 2-bit transformers called e-transformers .
The following subsections describe how an Edon80 part works in the three different modes.

3.1 Key Setup mode

When working in *Key Setup* mode, after transferring 80 bits in the register *Key*, the key is virtually divided into 40 2-bit consecutive values. That means we represent *Key* as *Key* = *K*0,

*K*1,…..,*K*39, where each *Ki* consists of 2 bits, and thus it can have a value from 0 to 3.
according to this values, in the *Key Setup* mode every working quasi group operation *i *;* i=0*;* 1, 2,…,79 is assigned by the following rules:

$$(Q; *i) \begin{cases} (Q; \bullet K_i) & 0 \le i \le 39 \\ (Q; \bullet K_{i-40}) & 40 \le i \le 79 \end{cases}$$

of the stream cipher is that the key stream consist of every second value that comes out from a_{79}.

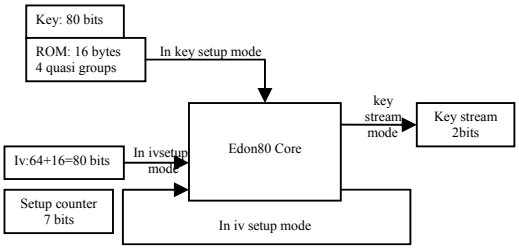

Fig 3.2 Edon80 components and

3.2 IV Setup mode

IV Setup mode in fact defines the initial values of the internal states $a_{0,\ldots,}a_{79}$, from the values of initial vector IV. Recall that the initial vector *IV* of length 64 bits is padded by 16 constant bits 1110010000011011, represented as 321001234 by 2-bits. Thus, the padded initial vector *IV* is a concatenation of 40 2-bit variables $IV = v_0,\ldots,v_{31},3,2, 1, 0, 0, 1, 2, 3 = v_0,\ldots,v_{39}$. Then we perform 80 e-transformations on *IV*.

After all 80 e-transformations are performed, the values of $a_{0,\ldots,}a_{79}$ are initialized by the following assignments: $a_i \leftarrow t_{79;i}$; i = 0,…,79

3.3 Key stream mode

To start the *Key stream* mode we just reset the value of *Counter* to 0 and set the value of T_0 to 1.In the *Key stream* mode we feed the *Edon80* Core by the values in the register *Counter* which increases its value every cycle. After a latency of 80 cycles, key stream starts to flow from the last e-transformer i.e. from the 2-bit register a_{79}.

4. Mickey

We present a stream cipher MICKEY (which stands for Mutual Irregular Clocking KEY stream generator) is aimed at resource-constrained hardware platforms. It is intended to have low complexity in hardware, while providing a high level of security. It uses irregular clocking of shift registers, with some novel techniques to balance the need for guarantees on period and pseudo randomness against the need to avoid certain cryptanalytic attacks.

MICKEY 2.0 takes two input parameters:

- An 80-bit secret key K, whose bits are labeled k_0 .. k_{79} ;
- An initialization variable IV, anywhere between 0 and 80 bits in length, whose bits are labeled $iv_{0,}\ldots,iv_{IVLENGTH-1}$.

The key stream bits output by MICKEY 2.0 are labeled Z_0, Z_1…. Cipher text is produced from plaintext by bitwise XOR with key stream bits, as in most stream ciphers.

Design principles including security aspects
The variable clocking of R:
When CONTROL _BIT _R=. 0 as shown in figure 4.1, the clocking of R is a standard linear feedback shift register clocking operation (with Galois-style feedback, following the primitive characteristic polynomial

$$c_R(X) = X^{100} + \sum_{I \in RTAPS} X^i \quad \text{eq. 4.1}$$

with INPUT_BIT _R XORed into the feedback). If we represent elements of the field GF (2^{100}) as polynomials

$$\sum_{i=0}^{99} r_i X^i \text{ , modulo } C_R(x) \text{ eq. 4.2}$$

then shifting the register corresponds to multiplication by x in the field.

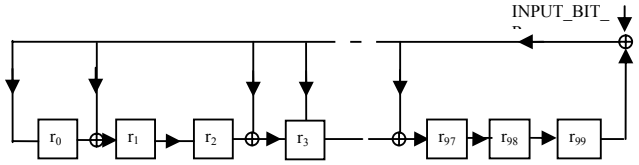

Figure 4.1 Clocking the R register with CONTROL_BIT_R=0

4.1 Key loading and initialization

The registers are initialized from the input variables as follows:

- Initialize the registers R and S with all zeros.
- (Load in IV)
  For $0 \le i \le IVLENGTH$ -1
  CLOCK_KG (R, S, MIXING =TRUE, INPUT_BIT = iv_i)
- (Load in K)
  For $0 \le i \le 79$:
  CLOCK_KG (R, S, MIXING = TRUE, INPUT_BIT = k_i)

- (Preclocking)
  For $0 \le i \le 99$
  CLOCK_KG (R, S, MIXING = TRUE, INPUT_BIT =0)

4.2 Generating key stream

Having loaded and initialized the registers, we generate key stream bits z0 to z_{L-1} as follows:

- For $0 \le i \le L-1$
  $z_i = r_0 \oplus s_0$
- CLOCK_KG (R, S, MIXING = FALSE, INPUT_BIT = 0)

5. Security of three stream ciphers

5.1 Trivium

Guess and Determine attacks

In each iteration of Trivium, only a few bits of the state are used, despite the general rule-of-thumb that sparse update functions should be avoided. As a result, guess and determine attacks are certainly a concern. A straightforward attack would guess ($s_{25,\ldots,}s_{93}$), ($s_{97,\ldots,}s_{177}$), and ($s_{244,\ldots,}s_{288}$), 195 bits in total, after which the rest of the bits can immediately be determined from the key stream**.**

Algebraic attacks

Trivium seems to be a particularly attractive target for algebraic attacks. The complete scheme can easily be described with extremely sparse equations of low degree.

Resynchronization attacks

Another type of attacks are resynchronization attacks, where the adversary is allowed to manipulate the value of the IV, and tries to extract information about the key by examining the corresponding key stream.

5.2 Security of Edon80

The internal state of *Edon80* has two parts:

1. Assignment of the working quasi groups *i = 0…39, which can be done in $2^{80} = 4^{40}$ ways.

2. Actual values of ai; i = 0……79 which has a space of 4^{80} possibilities. So, it follows that the total internal space of *Edon80* is $4^{40} * 4^{80} = {}_2{}^{240}$ Thus we can conclude that a simple attack by searching the state space is much worst then the exhaustive search attack on the key, which is 2^{80}. It says that exhaustive search attack or brute-force search is the best attack on *Edon80*.

Security on Related key attack

Related key attack is attempted to find two different keys that will produce the same key stream. *Edon80* uses initial values for $a_{0,\ldots,}$ a_{79} obtained by the *IV Setup* procedure where every bit of the key, is involved in highly correlated and nonlinear way.

Security of *IV Setup* mode

We would consider that the adversary would pose a threat to the security of the system if, by knowing the initialization vector *IV*, she/he can gain some knowledge about the internal states of the cipher.

5.3 The principles behind the design of MICKEY 2.0 are:

- To take all of the benefits of variable clocking, in protecting against many forms of attacks;
- To guarantee period and local randomness;
- Subject to those, to reduce the susceptibility to statistical attacks as far as possible.

The variable clocking architecture in MICKEY 2.0 is given in figure 5.8, the register R acts as the "engine", ensuring that the state of the generator does not repeat within the generation of a single key stream sequence, and ensuring good local statistical properties. The influence of R on the clocking of S also prevents S from becoming stuck in a short cycle.

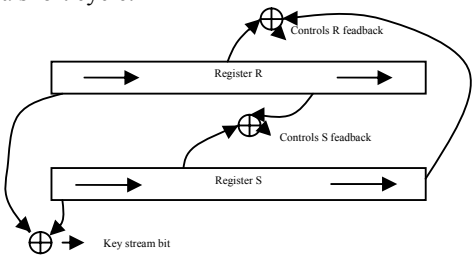

Figure 5.1 Variable clocking architecture

Key loading

We use a non-linear loading mechanism to protect against resynchronization attacks.

Algebraic attacks

Algebraic attacks usually become possible when the key stream is correlated to one or more linearly clocking registers, whose clocking is either entirely predictable or can be guessed.

We have taken care that the attacker cannot eliminate the uncertainty about the clocking of either register by guessing a small set of values. (By illustrative contrast, some attacks on LILI-128 [14] were possible because the state of the 39-stage

register could be guessed, and then the clocking of the 89-stage register became known.) Furthermore, each key stream bit produced by MICKEY 2.0 is not correlated to the contents of either one register (so in particular not to the "linear register" R).

6. Comparisons of three algorithms

In this chapter we will compare Trivium, MICKEY and Edon80 algorithms by giving the brief overview about their security issues

● Trivium is a simple synchronous stream cipher which seems to be particularly well suited for application which requires a flexible hardware implementation. Considering the security, there are two types of correlation attacks. First one is the correlation between key stream bits and its internal state bits.

- Edon80 is a new stream cipher that is proposed for hardware based implementations. Coming to the security, the internal space of *Edon80* is $4^{40} * 4^{80} = {}_2{}^{240}$, thus we can conclude that a simple attack by searching the state space is much worst then the exhaustive search attack on the key.

● MICKEY is intended to have low complexity in hardware, while providing a high level of security

	Trivium	**Edon80**	**MICKEY**
KEY/IV	80/80	80/64	80/80
Output Bits	1	2	1
LFSR Type	Nonlinear filter	------	Clock Controlled & Nonlinear Combiner
Internal State Space	2^{288}	2^{240}	2^{200}
Performance	$34*10^6$ bits/sec	$27*10^6$ bits/sec	$30*10^6$ bits/sec
Key Stream	2^{64}	------	2^{40}
Period	2^{93}	2^{103}	------

Table 6.1 Comparison of Trivium Edon80 and Micky in algorithmic perspective

7. Conclusion

Stream ciphers are a widely studied and used class of encryption algorithms, of which synchronous stream ciphers appear to offer the best combination of security and efficiency.

we have implemented three synchronous stream ciphers namely Trivium, Edon80 and Micky.

Apart from the implementation, we also studied these techniques against several stream cipher attacks by thoroughly analyzing the mathematical properties of their building blocks.

References

Prof. Dr. J¨org Keller, ***A Hardware-Based Attack on the A5/1 Stream Cipher,2001.***

[1.] ***Alfred Menezes, Paul van Oorschot, and Scott Vanstone. Handbook of Applied Cryptography. CRC Press, 1997.***

[2.] ***Daemen, J., Rijmen, V.: The Design of Rijndael: The Advanced Encryption Standard. Springer-Verlag (2002)***

[3]. ***Daemen. J.,Cipher and hash function design. Strategies based on linear and differential cryptanalysis. PhDthesis, Katholieke Universiteit Leuvfen ,1995***

[4.] ***E. Key, An analysis of the structure and complexity of nonlinear binary sequence generators. IEEE Transactions on Information Theory, 22: 732-736, 1976***

[5.] ***J.D. Golic, On the security of nonlinear filter generators. Fast software Encryption'96, volume 1039 of Lecture notes in computer science, 1996***

[6]. ***C.G. Gunther, Alternating step generators controlled by de Bruijin sequence, Advances in cryptology EUROCRYPT'87 volume 304, 1988.***

b. William Stallings, Cryptography and Network Security, Third Edition

[7]. ***F.Armknecht, A linearization attack on the Bluetooth key stream generator, Accessed September, 2003***

[9]. ***T.Siegenthaler, Correlation – immunity of non-linear combining functions for cryptographic applications. IEEE Transactions on Information Theory. 30:1984***

[10]. ***J.Dhem, F.Koeune, P. Lexour, A practical implementation of the timing attack. Technical Report CG-1998/1***

[11]. ***GOrodon Agnew, Thomas Beth, Arithmetic operations in GF(2^m). Journal of Cryptology -1993.***

[12]. ***J. Lano, N. Mentens, B. Preneel, and I. Verbauwhede, "Power Analysis of Synchronous Stream Ciphers with Resynchronization Mechanism," in ECRYPT Workshop, SASC (The State of the Art of Stream Ciphers).***

[13]. ***E.Dawson, A.clark,J.Golic, The LILI-128 Key stream generator ,First Open NESSIE Workshop -2000***