Jour. Pure Math., Vol. 31, 2014, pp. 77-87

IMPROVED MAC BASED DIFFERENTIAL FAULT ANALYSIS OF GRAIN-128a

PRAKASH DEY AND AVISHEK ADHIKARI*

ABSTRACT : Differential Fault Attack (DFA) on stream ciphers is an active field of research. However, only two differential fault attacks were reported on Grain-128a so far. Moreover, among these two, only the scheme proposed in Banik et al. [3] targeted the cipher with MACs corresponding to the massages chosen by the adversary. But the attack strategy of [3] required huge number of fault injections and invocations of MAC generation routines. To be specific, it required less than 2^{11} fault injections and invocations of less than 2^{12} MAC generation routines. In this current paper we propose an efficient MAC based DFA on Grain-128a. To the best of our knowledge, the proposed paper, for the first time shows that under certain situations the MAC generation mechanism of Grain-128a reveals all suppressed pre-output bits. Once the suppressed pre-output bits are obtained by the adversary, SAT solvers are used to obtain the secret key. Our proposed attack is achieved just by observing the correct and faulty MACs of certain chosen messages (MAC for empty message is not required) with no more than 55 fault injections and no more than 60 re-keying (invoking MAC generation routine each time) of the cipher device. This is a significant improvement over [3]. Moreover, by allowing random unknown single bit faults at both the LFSR and the NFSR, we relax the fault model as considered in [3].

Keywords : Stream Cipher, Differential Fault Attack, MAC, Grain-128a, SAT Solver.

1. INTRODUCTION

Grain-128a [2] is the successor of Grain-128 in the Grain family of ciphers. It has low gate count, a low power consumption and a small chip area. Authors of Grain-128a also claimed that Grain-128a offers better scurity than any existing 128 bit cipher with the added possibility of authentication. Further after the publication of Grain-128a, the eSTREAM finalist Grain-128 was no longer recommended. Mode of operation (i.e., whether authentication is mandatory or not) of Grain-128a depends on the first bit of the IV. Up to now, only two DFAs ([3] and

*Corresponding Author

[11]) were proposed on Grain-128a. In the DFA model, faults are injected into the internal state of the cipher and from the difference of the normal and the faulty outputs, information about the internal state is partially or completely deduced [6], [4], [5], [10], [11], [8], [7], [9]. Faults can be injected in a register by under-powring and power spike, clock glitch, temperature attack, optical attack, electromagnetic (EM) fault injection, etc. Effect of the first three methods can hardly be focused to a particular part of the device. On the other hand, optical and EM methods can affect a very restricted area. Banik et al. [3] proposed a DFA on Grain-128a that recovers the secret key using MACs. It requires less than 2^{11} fault injections and invocations of less than 2^{12} MAC generation routines. Also it was assumed that the cipher device can be re-keyed with the original key-IV or the original key and different IVs. Sarkar et al. [11] proposed a DFA on Grain-128a in which normal (fault free) and faulty keystreams (due to single bit faults) were used to recover the key-IV (and requires 10 faults). It was assumed that the cipher device can be re-keyed with the same key-IV. This paper proposes a DFA on Grain-128a using MACs only and assuming that the cipher device can be re-keyed with the same key-IV. In this case our attack strategy recovers the secret key and IV with no more than 55 faults and no more than 60 MAC generation calls.

2. DESCRIPTION OF GRAIN-128a

The Grain-128a [1, 2] cipher consists of a 128-bit non-linear feedback shift register (NFSR) X and a 128-bit linear feedback shift register (LFSR) Y. The NFSR and LFSR together represent the internal state of the cipher.

At round i (≥ -256), the internal state S_i of the Grain-128a cipher is given by,

$$S_i = (\underbrace{x_i, \ldots, x_{i+127}}_{X_i} \mid \underbrace{y_i, \ldots, y_{i+127}}_{Y_i}),$$

where $X_i = (x_i, \ldots, x_{i+127})$ and $Y_i = (y_i, \ldots, y_{i+127})$ respectively denote the inner states of X and Y.

Key Loading algorithm (KLA). The 128-bit secret *key* $(k_0, \ldots, k_{127})$ and 96-bit *IV* $(IV_0, \ldots, IV_{95})$ are used to initialize the initial state (at round $i = -256$) as follows:

$$S_{-256} = (\underbrace{k_0,\ldots,k_{127}}_{X} \mid \underbrace{IV_0,\ldots,IV_{95},1,\ldots,1,0}_{Y}).$$

Key scheduling Algorithm (KSA). During rounds $i = -256$ to -1 (called the KSA rounds), the registers X and Y are respectively updated by $x_{i+128} = z_i + f(X_i, Y_i)$ and $y_{i+128} = z_i + g(Y_i)$ where,

$f(X_i, Y_i) = y_i + x_i + x_{i+26} + x_{i+56} + x_{i+91} + x_{i+96} + x_{i+3}x_{i+67} + x_{i+11}x_{i+13} + x_{i+17}x_{i+18} + x_{i+27}x_{i+59} + x_{i+40}x_{i+48} + x_{i+61}x_{i+65} + x_{i+68}x_{i+84} + x_{i+88}x_{i+92}x_{i+93}x_{i+95} + x_{i+22}x_{i+24}x_{i+25} + x_{i+70}x_{i+78}x_{i+82}$,

$g(Y_i) = y_i + y_{i+7} + y_{i+38} + y_{i+70} + y_{i+81} + y_{i+96}$

and $z_i = h(X_i, Y_i)$ is given by,

$h(X_i, Y_i) = x_{i+2} + x_{i+15} + x_{i+36} + x_{i+45} + x_{i+64} + x_{i+73} + x_{i+89} + y_{i+93} + x_{i+12}x_{i+95}y_{i+94} + x_{i+12}y_{i+8} + y_{i+13}y_{i+20} + x_{i+95}y_{i+42} + y_{i+60}y_{i+79}$.

Here $f(X_i, Y_i)$, $g(Y_i)$ and $h(X_i, Y_i)$ are respectively called the update function of NFSR, the update function of LFSR and the pre-output function. The bit z_i is called the pre-output bit generated at the round i. During KSA rounds $i = -256$ to -1, the pre-output bit is fed back and XOR-ed with the input, both to the NFSR and to the LFSR. This pseudo-randomizes the internal state.

Pseudo-Random keystream Generation Algorithm (PRGA). After the KSA, in the PRGA rounds ($i \geq 0$), the pre-output bits are no longer XORed to the inputs of NFSR or LFSR and the registers X and Y are respectively updated by $x_{i+128} = f(X_i, Y_i)$ and $y_{i+128} = g(Y_i)$. The pre-output stream z produced at the PRGA rounds is $z_0, z_1, z_2, \ldots$ and is used for keystream and MAC generation (optional).

Modes of Operation. Grain-128a supports two different modes of operation: with and without authentication. Authentication is mandatroy when $IV_0 = 1$, and forbidden when $IV_0 = 0$.

Keystream Generation. With $IV_0 = 0$, the output function is defined as simply $w_i = z_i$ ($i \geq 0$), meaning all the pre-output bits $z_0, z_1, z_2, \ldots$ are used directly as keystream. With $IV_0 = 1$, the output function is defined as $w_i = z_{64+2i}$ ($i \geq 0$), meaning that the cipher picks every second bit as output of the cipher after skipping the first 64 bits. Those 64 initial bits and the other half will be used for authentication. In either case $w_0, w_1, w_2, \ldots$ represents the generated keystream. When $IV_0 = 1$, the pre-output bits $z_0, \ldots, z_{63}$ and $z_{65}, z_{67}, z_{69}, \ldots$ are all suppressed. This pre-out stream $z_0, \ldots, z_{63}$ and then $z_{65}, z_{67}, z_{69}, \ldots$ will be denoted by $(z_0, \ldots, z_{63}) \| (z_{65}, z_{67}, z_{69}, \ldots)$ and will be called the *suppressed keystream.* We will denote the suppressed pre-output bits by $(s_0, s_1, s_2, \ldots)$ where $s_i = z_i$, for $0 \le i \le 63$ and $s_i = z_{65+2i}$ for $i \ge 0$.

Thus we are considering three bit-streams, namely

1. the pre-output stream $z = (z_0, z_1, z_2, \ldots)$,
2. the normal keystream $w = (w_0, w_1, w_2, \ldots)$ and
3. the suppressed keystream $s = (s_0, s_1, s_2, \ldots)$.

MAC Generation Algorithm (MGA). We assume a message of length L defined by the bits $m_0, \ldots, m_{L-1}$. In this case $m_L = 1$ must be used as a padding. In order to provide authentication, two registers called the accumulator and the shift register of size 32 bits each, are used. The content of the accumulator at time t is denoted by $a_t^0, \ldots, a_t^{31}$. The content of the shift register is denoted by $r_t, \ldots, r_{t+31}$. The accumulator is initialized through $a_0^j = z_j$, $0 \le j \le 31$, and the shift register is initialized through $r_t = z_{32+t}$, $0 \le t \le 31$. The shift register is updated as $r_{t+32} = z_{65+2t}$ The accumulator is updated as $a_{t+1}^j = a_t^j + m_t r_{t+j}$ for $0 \le j \le 31$ and $0 \le t \le L$. The final content of the accumulator, $(a_{L+1}^0, \ldots, a_{L+1}^{31})$, is used for authentication.

Remark. One should note that, the stream $(z_0, \ldots., z_{31}, r_0, r_1, r_2, \ldots)$ is identical with the suppressed keystream $(s_0, s_1, s_2, \ldots)$ and $a_{L+1}^j = z_j + \sum_{t=0}^{L} m_t r_{t+j}$ for $0 \le j \le 31$. We denote the message $m_0, \ldots, m_{L-1}$ simply by $msg = m_0 \ldots m_{L-1}$ and the generated MAC corresponding to the message simply by $\sigma(msg) = (a_{L+1}^0, \ldots, a_{L+1}^{31})$. The empty message will be denoted by ϕ. Throughout the paper we shall now assume that $IV_0 = 1$ i.e., authentication is mandatory.

3. PROPOSED ATTACK ON GRAIN-128a : ATTACK MODEL, TOOLS AND DEFINITIONS

Notations. We shall use the following notations:

1. The empty set will be denoted by $\emptyset$.

2. For any two integers a and b, we denote the set $\{x : x \text{ is an integer with } a \leq x \leq b\}$ simply by $[a, b]$.

3. The i-th element u_i of any vector $u = (u_0, ..., u_{p-1})$ of length p will be denoted by $u(i)$, $\forall i \in [0, p-1]$.

4. For any two vectors $u = (u_0, ..., u_{p-1})$ and $v = (v_0, ..., v_{p-1}) \in \{0, 1\}^p$ of equal length, we denote $u + v = (u_0 + v_0, ..., u_{p-1} + v_{p-1})$.

5. For any vector $u = (u_0, .., u_{p-1}) \in \{0, 1\}^p$ and any bit $b \in \{0, 1\}$ we denote $bu = (bu_0, ..., bu_{p-1})$. Also We denote $1u$ simply by u.

6. For any two vectors $u = (u_0, ..., u_{p-1})$ and $v = (v_0, ..., v_{p-1}) \in \{0, 1\}^p$, we denote $u\|v = (u_0, ..., u_{p-1}, v_0, ..., v_{p-1})$. Thus $\|$ represents the concatenation operator.

Fault Location. A single bit fault flips a register bit value. We assume that all faults are injected at the beginning of the PRGA round 0. If a fault flips a register bit value then its position will be called as the fault location.

Attack model. In this paper we assume that the adversary is given access to an Oracle which possesses the unknown key and IV. The adversary queries the Oracle for the following information:

1. MACs $\sigma(msg_0)$, ..., $\sigma(msg_{p-1})$ corresponding to the p messages msg_0, ..., msg_{p-1} chosen by the adversary, all generated by the normal (fault free) pre-output stream z.

2. MACs $\sigma^{\phi}(msg_0)$, ..., $\sigma^{\phi}(msg_{p-1})$ corresponding to the same p messages msg_0, ..., msg_{p-1}, all generated by the faulty pre-output stream z^{ϕ}, for m (chosen by the adversary) fault locations $\phi \in \{\phi_0, ..., \phi_{m-1}\}$, where the m distinct fault locations $\phi_0, ..., \phi_{m-1}$ are chosen by the Oracle and are not given to the adversary.

In this case, m (chosen by the adversary) register locations are needed to disturb. A total mp number of faults are required to be injected and total $(m + 1)p$ re-keying are needed. The adversary only has $\sigma(msg_0), \ldots, \sigma(msg_{p-1})$ and $\sigma^{\phi}(msg_0), \ldots, \sigma^{\phi}(msg_{p-1})$, $\forall\phi \in \{\phi_0, \ldots, \phi_{m-1}\}$ for the messages $msg_0, \ldots, msg_{p-1}$ chosen by the adversary. Note that $\phi_0, \ldots, \phi_{m-1}$ are not known to the adversary. In this paper we want to minimize mp.

Attack strategy in a nutshell: With the information obtained from the Oracle, the adversary first finds the suppressed keystream bits by solving some algebraic equations, then identifies the fault locations $\phi_0, \ldots, \phi_{m-1}$. After that the adversary passes the available information to the SAT solver in SAGE and extracts the fault free internal state at round $i = 0$. Now by iterating backwards the key-IV can be recovered.

4. SUPPRESSED KEYSTREAM DETERMINATION

We now agree to denote the k-bit message $\underbrace{0\ldots0}_{k}$ simply by 0_k.

Let us consider the 4 messages 1, 0, 0_{32} and 0_{64}. Note that the corresponding MACs are available to the adversary.

One should note that,

$$\sigma(1) = (z_0 + r_0 + r_1, z_1 + r_1 + r_2, \ldots, z_{30} + r_{30} + r_{31}, z_{31} + r_{31} + r_{32}),$$

$$\sigma(0) = (z_0 + r_1, z_1 + r_2, \ldots, z_{30} + r_{31}, z_{31} + r_{32}),$$

$$\sigma(0_{32}) = (z_0 + r_{32}, \ldots, z_{31} + r_{63}),$$

$$\sigma(0_{64}) = (z_0 + r_{64}, \ldots, z_{31} + r_{95}).$$

Thus $\sigma(1)$ and $\sigma(0)$ give $r_0, \ldots, r_{31}$ and from $\sigma(0)$ we obtain $z_0, \ldots, z_{30}$.

Again from $\sigma(0_{32})$ and $\sigma(0)$ we obtain z_{31} and $r_{32}, \ldots, r_{63}$.

Similarly $\sigma(0_{64})$ and $\sigma(0_{96})$ give $r_{64}, \ldots, r_{127}$.

Thus we have $z_0, \ldots, z_{31}, r_0, \ldots, r_{127}$ i.e., we have the suppressed keystream bits $(z_0, \ldots, z_{63})||(z_{65}, z_{67}, z_{69}, \ldots, z_{255})$. And these bits are obtained without using the empty message ϕ.

The suppressed keystream bits for higher PRGA rounds (in a chunk of 32) could be obtained by observing the MACs of 0_{32L} for $L > 2$ iteratively.

5. FAULT SIGNATURE

Let for some fixed key-IV, $(s_0, \ldots, s_{n-1})$ be the suppressed keystream of length n. Let us consider a fault location ϕ. Let $(s_0^{\phi}, \ldots, s_{n-1}^{\phi})$ be the faulty suppressed keystream of length n under the fault at location ϕ.

In this case the bitwise XOR difference of the normal and faulty suppressed keystreams will be called the XOR differential suppressed keystream of length n and will be denoted by $d^{\psi,n} = (d_0^{\psi}, \ldots, d_{n-1}^{\psi})$.

We now define the following sets,

$$sig_{\phi}^{1} = \{i \in [0, n-1] : \Pr[d_i^{\psi} = 1] = 1\};$$

$$sig_{\phi}^{0} = \{i \in [0, n-1] : \Pr[d_i^{\psi} = 0] = 1\};$$

$$sig_{\phi}^{=} = \{\{i, j\} : i, j \in [0, n-1] \text{ and } \Pr[d_i^{\psi} + d_j^{\psi} = 0] = 1\}$$

$$sig_{\phi}^{\neq} = \{\{i, j\} : i, j \in [0, n-1] \text{ and } \Pr[d_i^{\psi} + d_j^{\psi} = 1] = 1\}$$

In this case $sig_{\phi} = (sig_{\phi}^{1}, sig_{\phi}^{0}, sig_{\phi}^{=}, sig_{\phi}^{\neq})$ will be called the signature of the fault location ϕ and mathematically represents the occurrence of some certain events under the fault ϕ. One could use large number of experiments in order to compute the fault signatures.

6. FAULT LOCATION IDENTIFICATION

Let at the online stage, a single bit fault is injected at an unknown location ψ. The adversary will use the following instruction for detecting fault location:

Obtain the XOR differential suppressed keystream $d^{\psi,n} = (d_0^{\psi}, \ldots, d_{n-1}^{\psi})$.

Compute, $support^1 = \{i \in [0, n-1] : d_i^{\psi} = 1\}$.

Compute, $support^0 = \{i\ [0, n-1] : d_i^{\psi} = 0\}$.

Compute, $pf^1 = \{\phi : \phi \in [0, 255] \text{ and } sig_{\phi}^1 \subseteq support^1\}$.

Compute, $pf^0 = \{\phi \in pf^1 : sig_{\phi}^0 \subseteq \text{support}^0\}$.

Compute, $pf^{=} = \{\phi \in pf^0 : sig_{\phi}^{=} \neq \emptyset \text{ and } \{i, j\} \in sig_{\phi}^{=} \Rightarrow d_i^{\psi} = d_j^{\psi}\} \cup \{\phi \in pf^0 : sig_{\phi}^{=} = \emptyset\}$.

Compute, $pf^{\neq} = \{\phi \in pf^{=} : sig_{\phi}^{\neq} \neq \emptyset \text{ and } \{i, j\} \in sig_{\phi}^{\neq} \Rightarrow d_i^{\psi} + d_j^{\psi} = 1\} \cup \{\phi \in pf^{=} : sig_{\phi}^{\neq} \neq \emptyset\}$.

Define, $pf = pf^{\neq}$.

The basic idea is to check whether the pre-computed pattern (signature) of a fault location occurs in the XOR differential suppressed keystream. If the pattern due to a fault location occurs in the XOR differential suppressed keystream, then it is a possible fault location, otherwise we reject it. It should be noted that from the construction it immediately follows that the actual fault location $\psi \in pf$. Now if pf is singleton then, ψ is uniquely determined.

Experimental Result. In 2^{20} trials, we generated key-IV randomly and simulated the injection of random single bit faults to the internal state at PRGA round 0. Experimental results show that, by considering the first 256 PRGA rounds, the fault location could be identified uniquely with a probability of 0.8.

7. STATE RECOVERY USING SAT SOLVERS

The adversary wishes to recover the internal state of the cipher at the PGRA round 0 and starts with the following information:

1. m fault locations, given by $\beta = (\phi_0, \phi_1, ..., \phi_{m-1})$.
2. normal (fault free) suppressed keystream $s = (s_0, ..., s_{n-1})$ of length n.
3. m faulty keystreams $s^0, ..., s^{m-1}$ each of length n, where s^i is the faulty suppressed keystream due the fault ϕ_i.

Let the fault free internal state at the PRGA round i (≥ 0) be $S_i = (X_i, Y_i)$ where $X_i = (x_i, ..., x_{i+127})$ and $Y_i = (y_i, ..., y_{i+127})$, the internal state at PRGA round 0 being $S_0 = (x_0, ..., x_{127}, y_0, ..., y_{127})$. We treat each x_i and y_i as variables and consider the PRGA rounds 0, ..., k. Corresponding to each pre-output bit z_i, we introduce two new variables x_{i+128} ($i \geq 0$) and obtain the following two equations: $x_{i+128} = f(X_i, Y_i), y_{i+128} = g(Y_i)$. We also consider the equation $z_i = h(X_i, Y_i)$ when z_i is a suppressed keystream bit.

Let us now consider the fault location ϕ_j. Since the cipher device is re-keyed before each fault injection, after the fault injection, if the faulty internal state at PRGA round i be S_i^j then at the targeted fault injection PRGA round 0 we have, $S_0^j(e) = S_0(e) + 1$, $\forall e \in \phi_j$ and $S_0^j(e) = S_0(e)$, $\forall e \in [0, 255] \setminus \{\phi_j\}$. Again corresponding to each pre-output bit z_i, we introduce two new variables x_{i+128}^j, y_{i+128}^j ($i \geq 0$) and obtain three more equations.

Now the system of polynomial equations are simply passed on to the SAT solver in SAGE for extracting solution for the variables $x_0, ..., x_{127}, y_0, ..., y_{127}$. Now by reversing backwards the secret key could be recovered.

Experimental results (a total of 100 experiments) show that, for $m = 11$, $k = 255$ (MACs corresponding to the messages 1, 0, 0_{32}, 0_{64} and 0_{96}) the state could be recovered with probability 1.0 in an average time of 84.36 seconds.

8. CONCLUSION

In this current paper we propose an efficient MAC based DFA on Grain-128a. To the best of our knowledge, the proposed paper, for the first time, shows that under certain situations the MAC generation mechanism of Grain-128a reveals all suppressed pre-output bits. Once the suppressed pre-output bits are obtained by the adversary, SAT solvers are used to obtain the secret key. Our proposed attack is achieved just by observing the correct and faulty MACs of certain chosen messages (MAC for empty message is not requred) with no more than 55 fault injections and no more than 60 re-keying (invoking MAC generation routine each time) of the cipher device. This is a significant improvement over [3].

ACKNOWLEDGEMENT

Research of the corresponding author is partially supported by National Board for Higher Mathematics, Department of Atomic Energy, Government of India, Grant No. 2/48(10)/2013/ NBHM(R.P.)/R&D II/695.

REFERENCES

1. Martin Ågren, Martin Hell, Thomas Johansson, and Willi Meier. A New Version of Grain-128 with Authentication. In *Symmetric Key Encryption Workshop 2011,* 2011.
2. Martin Ågren, Martin Hell, Thomas Johansson, and Willi Meier. Grain-128a: A New Version of Grain-128 with Optional Authentication. *International Journal of Wireless and Mobile Computing,* 5(1): 48-59, 2011.
3. Subhadeep Banik, Subhamoy Maitra, and Santanu Sarkar. A Differential Fault Attack on Grain-128a using MACs. In *Security, Privacy, and Applied Cryptography Engineering,* pages 111-125. Springer, 2012.
4. Subhadeep Banik, Subhamoy Maitra, and Santanu Sarkar. A Differential Fault Attack on the Grain Family of Stream Ciphers. In *Cryptographic Hardware and Embedded Systems-CHES 2012,* pages 122-139. Springer, 2012.
5. Subhadeep Banik, Subhamoy Maitra, and Santanu Sarkar. A Differential Fault Attack on the Grain Family under Reasonable Assumptions. In *Progress in Cryptology-INDOCRYPT 2012*, pages 191-208. Springer, 2012.
6. Alexandre Berzati, Cecile Canovas, Guilhem Castagnos, Blandine Debraize, Louis Goubin, Aline Gouget, Pascal Paillier, and Stephanie Salgado. Fault Analysis of GRAIN-128. In *Hardware-Oriented Security and Trust, 2009. HOST'09. IEEE International Workshop on,* page 7-14. IEEE, 2009.
7. Prakash Dey and Avishek Adhikari. Improved Multi-Bit Differential Fault Analysis of Trivium. In Willi Meier and Debdeep Mukhopadhyay, editors, *Pragress in Cryptology - INDOCRYPT 2014,* Lecture Notes in Computer Science, pages 37-52. Springer International Publishing, 2014.
8. Prakash Dey, Abhishek Chakraborty, Avishek Adhikari, and Debdeep Mukhopadhyay. Improved Practical Differential Fault Analysis of Grain-128. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition,* DATE '15, pages 459-464, San Jose, CA, USA, 2015. EDA Consortium.

9. Prakash Dey, Raghvendra Singh Rohit, and Avishek Adhikari. Full key recovery of ACORN with a single fault. *Journal of Information Security and Applications*, Elsevier Journal, pages–, 2016. To Appear.

10. Sandip Karmakar and Dipanwita Roy Chowdhury. Fault analysis of Grain-128 by targeting NFSR. In *Progress in Cryptology-AFRICACRYPT 2011*, pages 298-315. Springer, 2011.

11. Santanu Sarkar, Subhadeep Banik, and Subhamoy Maitra. Differential Fault Attack against Grain family with very few faults and minimal assumptions. *IEEE Transactions on Computers*, 64(6): 1647-1657, 2015.

Prakash Dey
Department of Pure Mathematics
University of Calcutta
Kolkata-700019
West Bengal, India.
E-mail : pd.prakashdey@gmail.com

Avishek Adhikari
Department of Pure Mathematics
University of Calcutta
Kolkata-700019
West Bengal, India.
E-mail : avishek.adh@gmail.com