ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Issue 7, July 2017

Vulnerability Analysis of Authenticated Encryption

G. M. Abisha Grace[1]
Assistant Professor, Electronics and Communication Engineering, Jeppiaar Institute of Technology, Chennai, India [1]

Abstract: Authenticated Encryption algorithm, AEGIS-128 is implemented in Field Programmable Gate Array (FPGA) of the Side-channel Attack Standard Evaluation Board (SASEBO-G) and in ATmega 328P microcontroller of the MultiTarget Victim Board. Encryption algorithms may be subjected to attacks if it is not properly implemented with countermeasures. So it is necessary to analyse the Authenticated Encryption algorithms providing both confidentiality and authenticity. In this paper, study of Authenticated Encryption with Associated Data (AEAD), AEGIS-128 algorithm, implementation of the algorithm in FPGA and microcontroller, vulnerability analysis of the implemented algorithm in two different hardware platforms and a countermeasure based on Rotating S-box Masking (RSM) for the AEGIS-128 algorithm implemented in ATmega 328P microcontroller is proposed. Vulnerability analysis, include derivation of power model for AEGIS-128 algorithm, capturing of power traces from the target device and comparing the measured power traces with the power model to retrieve the secret key.

Keywords: AEAD, AEGIS, DPA, FPGA RSM

I. INTRODUCTION

AEGIS is one of the algorithms submitted for Competition for Authenticated Encryption: Security, Applicability, and Robustness, (CAESAR). Authenticated Encryption with Associated Data (AEAD) as in [1] is a block cipher mode of operation. AEGIS-128 is an algorithm for the protection of Associated Data (AD) which makes it suitable for protecting network packets. Differential Power Analysis (DPA) is an attack where detailed knowledge of the attacked device is not required. SecretKey of the attacked device can be retrieved even if the power trace recorded is extremely noisy. In this paper, Rotating S-box Masking (RSM) as in [3] is the countermeasure technique adopted for AEGIS-128 algorithm.

II. AEGIS-128 ALGORITHM

128 bit key (K_{128}), 128 bit Initialization Vector (IV_{128}), and Associated Data (*AD*) of the algorithm is used to encrypt and authenticate a plaintext. The *AD* length and plaintext length are chosen to be less than 2^{64} bits. 2^{64} bits are divided into blocks of 128 bits. This individual blocks are used to update the state. According to the algorithm 128 bits of plaintext or *AD* is equivalent to one block of plaintext or *AD*. Based on the number of blocks of plaintext and *AD* the number of steps required to generate ciphertext and tag will vary. AEGIS-128 algorithm as in [6] consists of initialization, processing of AD, encryption and finalization phases.

A. Initialization of AEGIS-128

The initialization phase of AEGIS-128 is the phase of loading the Key(K_{128}) and Initialization Vector (IV_{128})along with constant ($Const_0$ and $Const_1$). The states are updated by the 10 steps of transformation function with message (m_i) input as combination of K_{128} and IV_{128}.

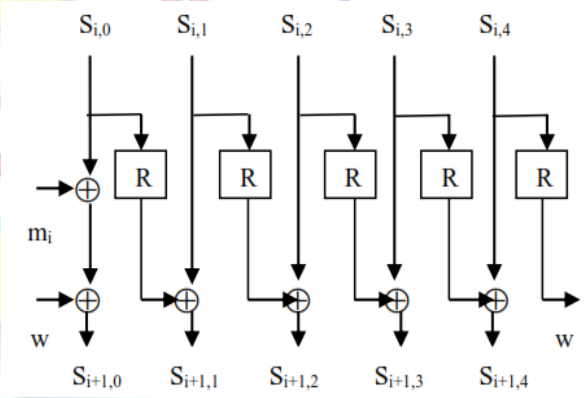


Fig. 1. StateUpdate

B. Processing of Associated Data of AEGIS-128

After the initialization phase, *AD* is used to update the states. If the last *AD* block is not a full block of 128 bits, zeros are used to make a block of 128 bits and the padded full block is used to update the state. If adlen=0, then the state will not get updated in this phase.

C. Encryption of AEGIS-128

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Issue 7, July 2017

After processing the *AD*, in each step of the encryption, a 16-byte plaintext block P_i is used to update the state and P_i is encrypted as ciphertext, C_i. If the last plaintext block is not a full block, zeros are used to make a full block of 128 bits and the padded full block is used to update the state. If msglen=0 then the state will not get updated in this phase.

D. Finalization of AEGIS-128

After encrypting the plaintext, the authentication tag is generated by performing the XOR operations.

III. IMPLEMENTATION OF AEGIS-128

AEGIS-128 algorithm has StateUpdate function that performs the single round transformations of AES as in [4] (S-box as in [2], ShiftRows and MixColumns as in [7]) five times. On the whole to process one block of *AD* and plaintext the single round transformation of AES is processed ninety five times by the AEGIS-128 algorithms

A. Serial Implementation

AES round transformation block containing S-box as in [5], ShiftRows and MixColumns[8] is processed five times serially by implementing it only once instead of implementing it five times for updating the 80 bytes of states. This is done by using a separate counter (Rcount) for performing the StateUpdate function and using temp register for storing the intermediate values as shown in Fig. 1.

B. Parallel Implementation

Five of AES round transformation blocks are processed simultaneously (80 bytes of states are updated in a single clock cycle). Input is given to the StateUpdate function where AES round transformations are performed simultaneously. This StateUpdate function is used for initialization, processing of *AD*, encryption and finalization of AEGIS-128 algorithm. AEGIS-128 algorithm using this parallel implementation in FPGA takes 20 clock cycles to generate tag. However, the area overhead increases according to the number of parallel operations. One AES round transformations block consists of S-box [9], ShiftRows and MixColumns.

IV. FPGA REALIZATION OF AEGIS-128

The program for control FPGA and cryptographic (or target) FPGA (Virtex 2-pro) is fed to the SASEBO-G board using the configuration cable with the help of Xilinx 9.1i from Personal Computer (PC). Light Emitting Diodes (LEDs) glow (these LEDs are active LOW) indicating the sequence of operations. The trigger signal will be active HIGH only if the encryption is enabled. This trigger signal is given as the input for the oscilloscope. Power consumption of the encryption operations performed by the cryptographic FPGA is displayed by the oscilloscope. Finally, power traces obtained from the oscilloscope is stored in the PC through the Local Area Network (LAN) cable. Ciphertext output from RS232 cable is displayed in Matrix Laboratory (MATLAB). 128-bit data is streamed as 8-bit input into the FPGAs. Once the encryption is complete and when the ciphertext and tag are available Output_rdy signal is set HIGH. This indicates the availability of 128-bit ciphertext in the data-bus as bytes which are then transferred through RS232 cable to the PC.

V. DPA ATTACK OF AEGIS-128 ALGORITHM IMPLEMENTED IN FPGA

Based on different attack scenarios twenty thousand different random plaintexts, IV_{128} and *AD* are generated. Power traces are captured and stored in MATLAB while running the twenty thousand encryption runs, as in Fig. 2. Since the key size is 128 bits, the search complexity is 2^{128} bits. In order to avoid this huge search space, 128-bit key is broken into 16 bytes and the attack is performed on each individual byte sixteen times to reveal all the 128 bits of the secretkey. The key complexity is reduced to 2^8 (256) bytes. Figure 2 shows the power trace obtained for 95 (19 StateUpdate with each StateUpdate having 5 AES round transformations) AES round transformations.

A. Differential Power Model and Statistical Analysis For AEGIS-128 Algorithm

1) Pre-initialization Attack
   a. Different and Known Plaintext, P[1:20,000]
   b. Different and Known IV, P[1:20,000]
   c. Different and Known Plaintext, P[1:20,000]

For 1st eight bit, the intermediate value chosen is $K_1 \oplus IV_1$, where K_1 and IV_1 are the 1st eight bits K_{128} and IV_{128} respectively. The point of attack is the 8-bit register a_0 holding the value $(K_1 \oplus IV_1)$. As a_0 register is initialized to zero for each encryption run Hamming-weight power model is used as in Eq. (1). Refer Fig. 3 for key guess. Suffix 1 to 16 indicates 1 to 16 bytes.

$$H = \text{Hamming-weight } [K_1 \oplus IV_1] \quad (1)$$

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Issue 7, July 2017

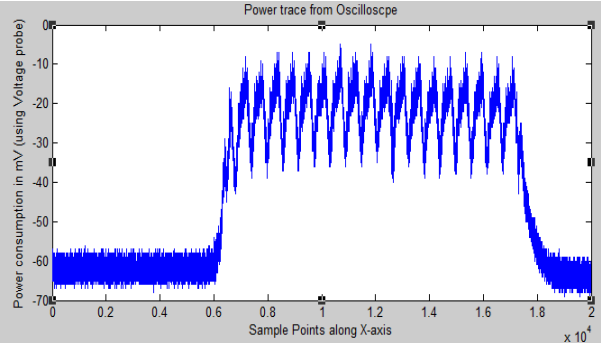


Fig. 2. Power trace for AEGIS-128 algorithm

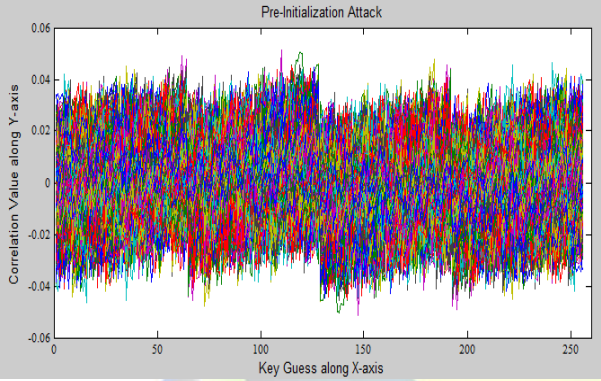


Fig. 3. Correlation value along Y-axis and Key guess along X-axis

The same procedure is repeated for all 16 bytes. As the attack is performed on 8 bits of the 768 bits (6 registers of each 128 bits), the Signal to Noise Ratio (SNR) (0.1065) is very low and the correct key is not retrieved. (SNR= $\frac{a}{n}$ = $\frac{8}{760}$, 'a' represent the number of wires processing a bit of attacked intermediate result and 'n' represent the number of wires processing statistically independent bits.)

2) Known Plaintext Attack

a. Different and known plaintext, P [1:20,000]]
b. Constant and known IV_{128} [1:20,000]
c. Constant and known AD [1:20,000]
d. Attack point: 2nd 128 bit of the first StateUpdate in the tag generation phase
e. $S_{1,0}$ stored in a_0 register is a known constant, the output of the processing of associated data phase (1st 128 bits of the 640 bits of the updated state).
f. SI is a known constant state, obtained as a result of XOR operation of $S_{1,0}$ with w_1 refer Fig. 4. ($S_{2,0} = SI \oplus P = A$).
g. $S_{2,1}$=Known constant state stored in temporary a_1 register. Refer Fig. 5.

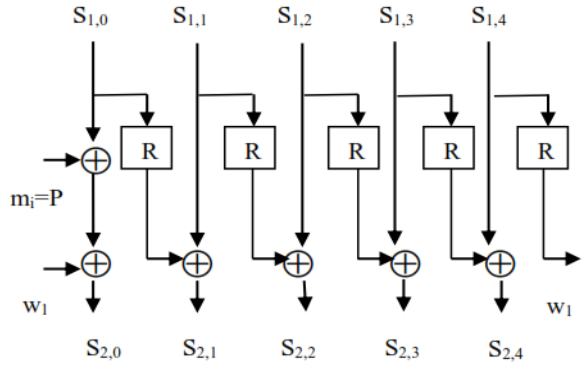


Fig. 4. StateUpdate of Encryption Phase

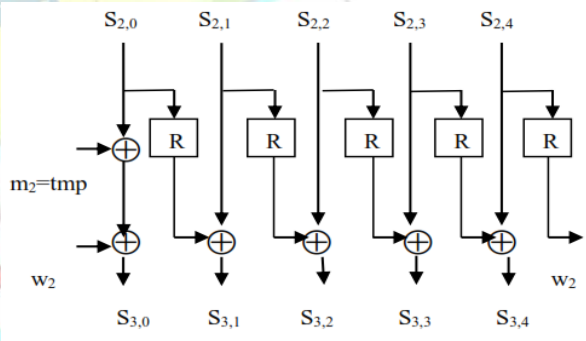


Fig. 5. Point of attack at the beginning of tag generation phase

To attack the 1st eight bits of the unknown state $S_{3,1}$, the intermediate value chosen is 1st byte of $S_{3,1}$ given as $[(02(S\text{-}box(SI_1 \oplus P_1))) \oplus (03(S\text{-}box(SI_6 \oplus P_6))) \oplus (01(S\text{-}box(SI_{11} \oplus P_{11})) \oplus (01(S\text{-}box(SI_{16} \oplus P_{16}))) \oplus B_1]$

B_1 value is obtained from the 1st byte of the known constant value stored in the temporary register, $a_1 = S_{2,1}$. Refer Fig. 6 for state guess. Hamming-Distance power model is used to determine the intermediate state as given in Eq. (2).

$$H = \text{Hamming-Distance between } SI_1 \oplus P_1 \text{ and } [(02(S\text{-}box(SI_1 \oplus P_1))) \oplus (03(S\text{-}box(SI_6 \oplus P_6))) \oplus (01(S\text{-}box(SI_{11} \oplus P_{11}))) \oplus (01(S\text{-}box(SI_{16} \oplus P_{16}))) \oplus B_1] \quad (2)$$

It is concluded that the algorithm is secure up to 20,000 power traces because the attack complexity is 2^{40}

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Issue 7, July 2017

computations (A_1, A_6, A_{11}, A_{16} and B_1 each of 8 bits) from Eq. 31 and the state is not revealed for 20,000 power traces. Attack complexity for 2nd byte is 2^{48} computations (A_1, A_2, A_6, A_{11}, A_{16} and B_1 each of 8 bits). Attack complexity varies for each byte based on the dependency of the states to be attacked. Attack complexity for the bytes 1, 5, 9 and 13 are 2^{40} computations. Attack complexity for the bytes 2, 3, 4, 6, 7, 8, 10, 11, 12, 14, 15 and 16 are 2^{48} computations. SNR is calculated as 0.01065.

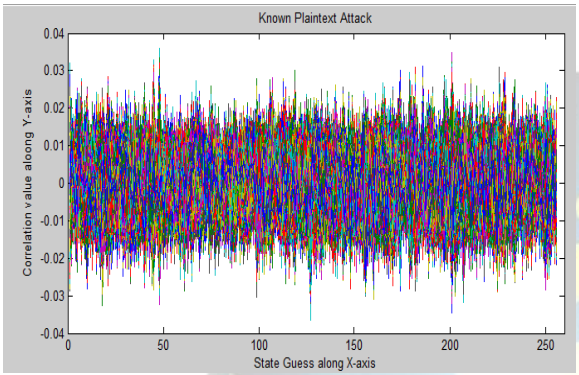


Fig. 6. Correlation value along Y-axis and State Guess along X-axis

VI. Vulnerability Analysis Of Aegis-128 In Microcontroller

A. *Microcontroller Implementation of AEGIS-128*

The entire AEGIS-128 algorithm is implemented in ATmega 328P microcontroller of the MultiTarget Victim Board. S-box, ShiftRows and MixColumns are implemented as functions in C language.

Memory required:

1) Program: 3830 bytes <11.7% Full>
2) Data: 416 bytes <20.3% Full>

The power trace for a single encryption run of the entire AEGIS-128 algorithm is shown in Fig. 7.

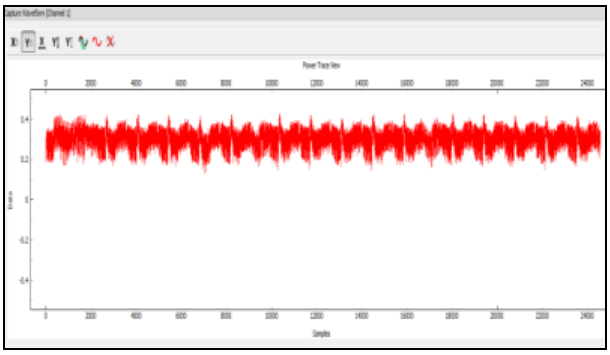


Fig. 7. Power in µW along Y-axis and Sample points along X-axis

B. *Leakage Power Model*

Attack point is at the initialization phase, refer Fig. 1 ($K_1 \oplus IV_1$) is passed through S-box and by the architecture of the microcontroller it is stored in a register. Hamming-weight power model is used to attack the key as given in Eq. (3).

$$H=Hamming\text{-}weight[S\text{-}box(K_{128} \oplus IV_{128})]. \quad (3)$$

As 128-bit key is partitioned into 16 bytes the key complexity is reduced to 256 bytes. For attack, trigger is given only to the first StateUpdate (Five AES rounds alone) to get more samples.

1) Random IV_{128} [d=1:200]
2) Fixed key, K_{128}

The matrix H of size d×k (200×256) is formed. Trace matrix obtained is d×t (200×16000) where t- sample points chosen. The correlation matrix R is formed by comparing the matrix T and the matrix H. The correct key is obtained within 30 power traces.

VII. Countermeasure of AEGIS-128 Algorithm

Rotating S-box Masking (RSM) countermeasure as in [3] is adopted for AEGIS-128 algorithm. The 16 byte mask chosen is

$$M = [03 \;\; 0C \;\; 35 \;\; 3A \;\; 50 \;\; 5F \;\; 66 \;\; 69 \;\; 96 \;\; 99 \;\; A0 \;\; AF \;\; C5 \;\; CA \;\; F3 \;\; FC]$$

Hamming weight distribution (0, 0, 4/16, 0, 8/16, 0, 4/16, 0, 0) of each byte is in the range of [2 4 6] instead of 0 to 8 so that SNR is reduced to 50% making it difficult to recover the mask. From M sixteen different 16-byte masks M_i (0<i<15) are calculated by left rotation of the bytes of M sixteen times. Corresponding 16 masked S-boxes are calculated and stored in flash memory. For Eq. (1), IV_{128} is masked by M_i by the XOR operation and then the resultant masked IV_{128} with K_{128} is fed as input to 16 different masked S-boxes. Masked S-box is calculated by the Eq. (4) and Eq. (5)

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Issue 7, July 2017

$$x = IV_{128} \oplus K_{128} \quad (4)$$

$$S-box(x \oplus m_i) = S-box(x) \oplus m_{i+1} \quad (5)$$

m_i and m_{i+1} denotes any two consecutive bytes of the matrix M called as old and new mask respectively. Masked S-box is calculated such that the positions of the S-box[10] are XOR with m_i and the values of the S-box are XOR with m_{i+1}. During each encryption run, a random offset is generated (0<i<15). Based on this offset, M_i is used as the mask and the resultant value is send to the corresponding masked S-boxes as in Eq. (5). The masked value undergoes ShiftRows and MixColumns, the result is shown in Eq. (6).

$$MixColumns[ShiftRows(S\text{-}box(y \oplus m_i))] = MixColumns\ ShiftRows(S\text{-}box(y))] \oplus MixColumns\ [ShiftRows\ (m_{i+1})] \quad (6)$$

y takes any value depending on the input to AES round transformation. The mask is removed during XOR operation of each StateUpdate by giving the Eq. (7) as input to XOR with Eq. (6)

$$z = MixColumns[ShiftRows(m_{i+1})] \quad (7)$$

After removing Eq. (7) from the Eq. (6), Eq. (6) becomes as in Eq. (8).

$$MixColumns[ShiftRows(S\text{-}box(y))] \quad (8)$$

In case of AEGIS-128, XOR operation in Fig. 1 due to masking carries the value of old mask m_i from the input (y_1) which does not undergo AES round transformations. The value stored in register has to be kept always in a masked state so while performing the XOR operation as shown in Fig.1. Eq. (7) and Eq. (8) are XOR along with the removal of the old mask m_i and the addition of the new mask m_{i+1} as in Eq. (9).

$$\{MixColumns[ShiftRows(S\text{-}box(y))]\} \oplus \{MixColumns[ShiftRows(m_{i+1})]\} \oplus \{MixColumns[ShiftRows(m_{i+1})]\} \oplus (y_1 \oplus m_i) \oplus m_i \oplus m_{i+1} = \{MixColumns[ShiftRows(S\text{-}box(y))]\} \oplus y_1 \oplus m_{i+1} \quad (9)$$

This is the main difference in unmasking between AES and AEGIS. Memory consumed for AEGIS-128 algorithm implemented using RSM is

1) Program: 29152 bytes <89.0% Full>
2) Data: 422 bytes <20.6% Full>

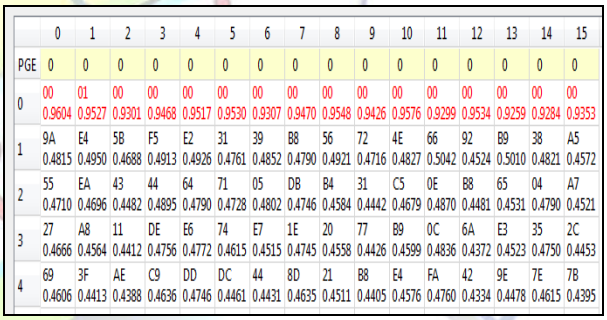
VIII. RESULTS AND DISCUSSIONS

AEGIS-128 algorithm is optimally implemented in FPGA by using serial implementation. Power traces are analysed to retrieve the secretkey. Due to architectural differences the algorithm implemented in FPGA is more secure than in microcontroller implementation. Area, speed and power for parallel and serial implementations of the algorithm in FPGA are discussed in Table 1.

TABLE I
TRADE-OFF SUMMARY

Summary	AEGIS-128 using parallel implementation for state update		AEGIS-128 using serial implementation for state update	
Area	*Used*	*Utilization*	*Used*	*Utilization*
No. Of Occupied slices	4,926	99%	2,749	55%
No. Of 4 i/p LUT	9,629	97%	3,927	39%
Throughput	1.958 Gbps		1.707 Gbps	
Power	84 mW		83 mW	

For AEGIS-128 algorithm implemented in microcontroller, the matrix R having the largest correlation value corresponds to the correct key and once the key is found it is available in the first row in the Fig. 8.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PGE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	00 0.9604	01 0.9527	00 0.9301	00 0.9468	00 0.9517	00 0.9530	00 0.9307	00 0.9470	00 0.9548	00 0.9426	00 0.9576	00 0.9299	00 0.9534	00 0.9259	00 0.9284	00 0.9353
1	9A 0.4815	E4 0.4950	5B 0.4688	F5 0.4913	E2 0.4926	31 0.4761	39 0.4852	B8 0.4790	56 0.4921	72 0.4716	4E 0.4827	66 0.5042	92 0.4524	B9 0.5010	38 0.4821	A5 0.4572
2	55 0.4710	EA 0.4696	43 0.4482	44 0.4895	64 0.4790	71 0.4728	05 0.4802	DB 0.4746	B4 0.4584	31 0.4442	C5 0.4679	0E 0.4870	B8 0.4481	65 0.4531	04 0.4790	A7 0.4521
3	27 0.4666	A8 0.4564	11 0.4412	DE 0.4756	E6 0.4772	74 0.4615	E7 0.4515	1E 0.4745	20 0.4558	77 0.4426	B9 0.4599	0C 0.4836	6A 0.4372	E3 0.4523	35 0.4750	2C 0.4453
4	69 0.4606	3F 0.4413	AE 0.4388	C9 0.4636	DD 0.4746	DC 0.4461	44 0.4431	8D 0.4635	21 0.4511	B8 0.4405	E4 0.4576	FA 0.4760	42 0.4334	9E 0.4478	7E 0.4615	7B 0.4395

Fig. 8. Results table displayed in ChipWhisperer Analyzer

IX. CONCLUSION

AEGIS-128 algorithm is studied and implemented in FPGA using AES rounds as a sub-module. AEGIS-128 algorithm is implemented in two ways. It is concluded that serial implementation occupies less area and consumes less power and takes more clock cycles than parallel implementation. Power traces are captured using oscilloscope. Power model for the AEGIS-128 algorithm is derived by analysing the known, unknown constants and variables. In FPGA implementation neither the key nor the intermediate states are obtained with 20,000 power traces. It is concluded that the algorithm is secure for 20,000 power traces. SNR for both the attack scenarios is 0.01065. Attack

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Issue 7, July 2017

complexity in pre-initialization attack is 2^8 computations attack complexity for known plaintext attack is 2^{40} computations for 1st, 5th, 9th and 13th bytes and 2^{48} computations for 2nd, 3rd, 4th, 6th, 7th, 8th, 10th, 11th,12th, 14th, 15th and 16th bytes.

Power trace of the AEGIS-128 algorithm implemented in ATmega 328p microcontroller is captured using ChipWhisperer. In the microcontroller platform as the intermediate results are stored in registers, the algorithm is attacked within 30 power traces and the attack complexity is 2^{12} computations. RSM is proposed as a good countermeasure for AEGIS-128 algorithm implemented in ATmega 328P microcontroller and the attack for the countermeasure is not successful for 1000 power traces.

In future, for performing the attack on the AEGIS-128 algorithm implemented in FPGA of the SASEBO-G board the number of traces can be increased and the attack can be performed to retrieve the key. Parallel implementation of AEGIS-128 algorithm can be taken to perform DPA attacks. Similarly template attacks can be tried on the AEGIS-128 algorithm implemented with RSM countermeasure in the ATmega 328P microcontroller of the MultiTarget Victim Board.

ACKNOWLEDGMENT

Society for Electronics Transaction and Security (SETS) is a leading institution in hardware security research. We thank SETS for offering the internship, where we acquired knowledge on CAESAR candidate. All the experiments and results reported in this paper were performed at SETS with guidance of the researchers of Hardware Security Research Group, SETS.

REFERENCES

[1]. M. Agren, M. Hell, T. Johansson, W. Meier, *Grain-128a: A New Version of Grain-128 with Optional Authentication*. International Journal of Wireless and Mobile Computing 2011, Vol. 5, No. 1 pp. 48-59.

[2]. Akashi Satoh, Sumio Morioka, Kohji Takano, and Seiji Munetoh, *A Compact Rijndael Hardware Architecture with S-box Optimization*, ©_ Springer-Verlag Berlin Heidelberg. C. Boyd (Ed.): ASIACRYPT 2001, LNCS 2248, pp. 239–254, 2001.

[3]. Alexander DeTrano, Naghmeh Karimi, Ramesh Karri, Xiaofei Guo, Claude Carlet and Sylvian Guilley, *Exploiting Small leakages in Masks to Turn a Second-Order Attack into a First-Order Attack and Improved Rotating Substitution Box Masking with Linear Code Cosets*, published in Hindawi Publishing Corporation at The Scientific World Journal, Volume 2015, Article ID 743618, 10 pages.

[4]. A. Bogdanov, F. Mendel, F. Regazzoni, V. Rijmen, and E. Tischhauser. *ALE: AES-Based Lightweight Authenticated Encryption. Fast Software Encryption* - FSE 2013.

[5]. D. Canright, *A Very Compact S-box for AES*, in Proc. Cryptographic hardware and Embedded Syst., Edinburgh, U.K., Sep. pp.441-455.

[6]. Hongjun Wu, and Bart Preneel, *AEGIS: A Fast Authenticated Encryption Algorithm1 (Full Version*), the original paper was published at Selected Areas in Cryptography (SAC 2013).

[7]. Hua Li and Zac Friggstad, *An Efficient Architecture for the AES Mix Columns Operation*, published in IEEE conference Publications, Circuits and systems pp. 4637 - 4640 Vol. 5

[8]. T. Itoh and S. Tsujii, *A Fast Algorithm for Computing Multiplicative Inverses in GF(2^m) using Normal Bases*, *Information and Computation*, Vol.78, No. 3, pp. 171–177.

[9]. Sumio Morioka and Akashi Satoh, *An Optimized S-box Circuit Architecture for LowPower AES Design*, .© Springer-Verlag Berlin Heidelberg. pp. 172–186, 2003.

[10]. Xinmiao Zhang, and Keshab K.Parhi, *On the optimum constructions of composite field for the AES algorithm*, IEEE transactions on circuits and systems-II:Express briefs,Vol.53,No.10,October2006.