International Journal of Engineering Research and General Science Volume 4, Issue 3, May-June, 2016
ISSN 2091-2730

Implementation of AES-192 Algorithm to Overt Fake Keys against Side Channel Attacks

Savitha.S[1], Asst.prof.Yamuna.S [2]

[1,2]Deparment of Electronics and Communication Engineering,

[1,2]Sri Shakthi Institute of Engineering and Technology, Coimbatore.

[1]ssavitha1993@gmail.com, 97550419947

Abstract— Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication and data origin authentication. In data and telecommunications, cryptography is necessary when communicating over any unreliable medium, which includes any network particularly the internet. In this paper, an approach to overt the cryptographic key with a fake key, when there happens any counter attack so-called side-channel attack (SCAs) is applied to break the security of AES-192. SCAs happens based on the correlation produced by the data provided as input and operations performed by the device for the data provided and its actual power consumption for the operation. The proposed approach revealing false key introduces few modifications in the existing AES algorithm which aims at masking the true key by reinforcing the correlation coefficient in such a way that the attack leads to a false key which misleads the attacker. The algorithm is coded in Verilog and simulated using Modelsim DE6.5e. The synthesis process is done using Xilinx ISE8.1i and implemented in Xilinx Spartan 3E device. Experimental Matlab R2014a results show the strength of the proposed system, which is capable of successfully hiding the true cryptographic key.

Keywords— Advanced Encryption Standard Encryption Standard, Data Side-channel attacks, Cryptography, cipher text, SubByte, ShiftRow, MixColumn, AddRound key

I. INTRODUCTION

In the late 1990s, Kocher et al. [1] proposed a simple methodology to obtain the key of a cryptographic algorithm which was based on analyzing the power consumption associated with the hardware device used in the implementation of such an algorithm. The information related to the revealed key through power consumption is used to perform the Side-Channel Attacks (SCAs). In addition to their conceptual simplicity, these attacks only need cheap instrumentation equipments used for capturing and data processing. Although authors implemented their proposal on a Data Encryption Standard (DES) algorithm, nowadays the process has been successfully applied over different cryptographic algorithms of private key [2]. However due to the high level of security, most publications have intended to find out the key of the popular Advanced Encryption Standard (AES) algorithm. Since its adoption by the NIST [13] as standard, this algorithm has significantly increased its popularity.

Hence this forms the basis for encrypting many official documents by the National Security Agency (NSA) and by the EUA government itself [3]. Many techniques have been developed to protect the integrity of cryptographic systems due the advent of SCAs. Counter measures that has been proposed for hiding leads to design a systems where the data processed and the power consumed are independent[12]. Such a system is achieved either by creating systems that are featured with random power consumption or systems whose power consumption is constant for every clock cycle [2].Another type of countermeasures called masking techniques was proposed. The robustness of such techniques is based on masking the data with random values that are unknown by the hackers. These Masked data change the power consumption of the masked data from the original data. It is to be noted that the functionality of the algorithm is not changed at any point of the system execution.

The countermeasures based on masking or hiding for Different versions of AES have been proposed and implemented in the past. Most of these proposals are based on implementations performed on microprocessors, ASIC or FPGAs [4] [5]. All these proposals are always addressed to design systems in which all the permitted keys that have been obtained upon a SCAs analysis are equally likely hence the concept of equiprobability is the real measure of protection is adopted. This paper presents a completely different approach towards masking of data that includes, instead of protecting the equiprobability of keys, the countermeasure is based on designing the

algorithm in such a way that a false key becomes the key with the highest probability obtained in a SCA by the attacker[6][7]. In fact, the system seems to behave as an unprotected system but when an attacker performs a classical SCA then the system culminates in a false positive [8].

The paper is organized in five sections. Section I describes about the evolution of AES. Section II describes the theoretical basis adopted in SCA and the existing work using AES-192. Section III shows proposed work of reveal false keys for AES-192. Section IV presents the experimental results, and finally section V shows the conclusions.

II. CLASSICAL STATISTICAL ATTACKS BASED ON POWER ANALYSIS

A. AES algorithm

Fig. 1 shows the general architecture of a crypto-processor system. A cryptographic algorithm is applied over a plain text in order to obtain a cipher text [10].

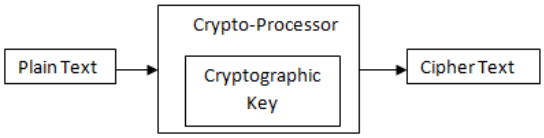

Fig.1 Crypto-processor architecture.

The brute-force attack, which tries to obtain the plain text without knowing the actual cryptographic key, is unfeasible. AES is a symmetric block cipher, so that it uses the same key for encrypting and decrypting both the plain and the cipher texts, respectively. The Advanced Encryption Standard is a block cipher whose design principle is known as a substitution permutation network. The operation of AES is performed on data of a fixed block length, using a key that can either have 128, 192, or 256 bits (this paper is based on a version of 192 bits). The AES algorithm with 128-bit, 192-bit and 256-bits are usually referred to as AES-128, AES-192 and AES-256 respectively. The key length represents the number of 32-bit words in the key, and thus is equal to 4, 6 or 8 in this standard. The input block, the output block and the intermediate cipher result all have the same length of 128 bits. The block size represents the number of 32-bit words in the block. The number of AES round is determined by its key size: 10 rounds for AES-128, 12 rounds for AES-192, and 14 rounds for AES-256[11].

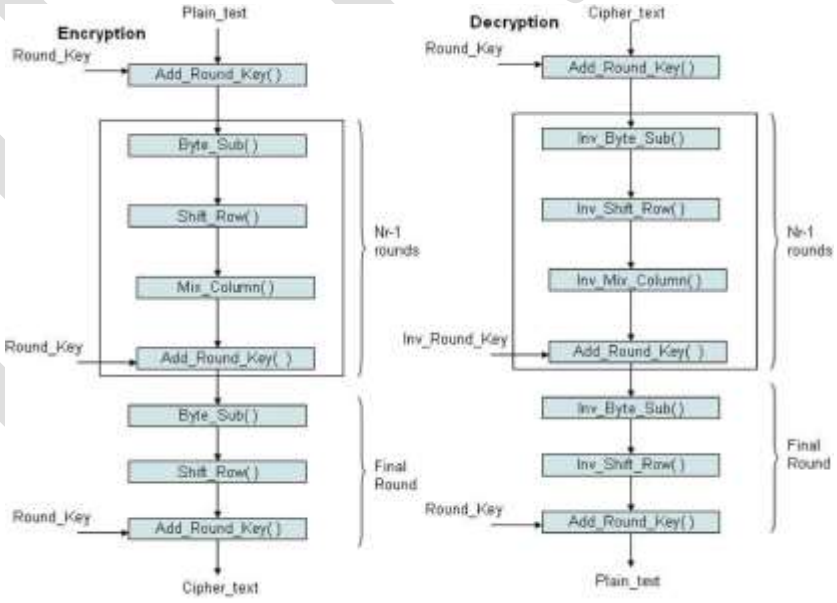

Fig.2. AES Encryption and Decryption

The algorithm applies several substitutions and permutations on a block of 128 bits, called the state, which initially is obtained by combining the plain text and the key by an exclusive-or operation [11]. Thus, the state is represented by an array of 16 bytes with four rows and four columns. Such functions are repetitively applied on the state during several iterations calls rounds. For both its Cipher

and Inverse Cipher, the AES algorithm uses a round function that is composed of four different byte-oriented transformations as shown in fig 2:

- Byte substitution using a substitution table (S-box),
- Shifting rows of the State array by different offsets,
- Mixing the data within each column of the State array, and
- Adding a Round Key to the State.

In each round, the state is again combined with a new key, called round key, which is created by the key scheduling algorithm (key expansion). A detailed description of the AES algorithm can be found in [3]. According to the theory of SCAs, the most vulnerable blocks that can be attacked are *AddRoundKey* and *SubBytes*. The AddRoundKey block is a simple bitwise XOR operator, whereas the SubBytes block is a non-linear function that is applied to each individual byte of the state.The*SubBytes* function can be implemented by means of a look-up table of 256 elements, which is called 16 times (one time for each byte of the state) per round.

B. AES Encryption

In encryption mode, the initial key is added to the input value at the very beginning, which is called an initial round. This is followed by 11 iterations of a normal round and ends with a slightly modified final round. During one normal round the following operations are performed in the following order: Sub Bytes, Shift Rows, Mix Columns, and Add Round key [11]. The final round is a normal round without the Mix Columns stage.

Steps in AES Encryption

- Sub Bytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
- Shift Rows—a transposition step where each row of the state is shifted cyclically a certain number of steps.
- Mix Columns—a mixing operation which operates on the columns of the state, combining the four bytes in each column
- Add Round Key—each byte of the state is combined with the round key; each round key is derived from the cipher key using a key schedule

The Sub Bytes transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box). This S-box which is invertible is constructed by composing two transformations which include an inverse function GF(2^8) and an invertible affine transformation. Conventionally, the coefficients of the SBox and inverse S-Box are stored in the lookup tables, or a hard wired multiplicative inverter over GF(2^8) can be used, together with an affine transformation circuit as in Fig.3.

In the Shift Rows transformation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row is not shifted at all, the second row is shifted by one the third row by two, and the fourth row by three bytes to the left. This has the effect of moving bytes to lower positions in the row (i.e., lower values of *c* in a given row), while the lowest bytes wrap around into the top of the row (i.e., higher values of *c* in a given row)[11]. The MixColumns function applies a linear transformation to the state, and it operates on the matrix column by column. Each column is treated as a polynomial over GF(2^8) and multiplied modulo x^4+1 with a fixed polynomial. The AddRoundKey transformation adds a round key to the state by using a simple bitwise XOR operation. Each round key used is derived from the secret key by employing the key schedule.

C. AES Decryption

In decryption mode, the operations are in reverse order compared to their order in encryption mode. Thus it starts with an initial round, followed by 11 iterations of an inverse normal round and ends with an AddRoundKey. An inverse normal round consists of the following operations in this order: AddRoundKey, InvMixColumns, InvShiftRows, and InvSubBytes. An initial round is an inverse normal round without the InvMixColumns. The Inverse Sub Bytes (InvSubByte) function applies the inverse S-box to each byte of the state. The operation is carried out by inverting the affine transformation and then taking the multiplicative inverse in GF(2^8). The Inverse Shift Rows Transformation (InvShiftRows) operation, the first row is unchanged, and row I is shifted to the right i byte(s) cyclically, where i=1, 2, or 3. The InvMixColumns is the inverse of MixColumn operation. Each column of the state is multiplied modulo x^4+1 with a fixed polynomial. The AddRoundKey transformation is its own inverse as it only involves the xor operation [11].

D. Fundamentals of statistical attacks

SCAs exploit the existing correlation between the power consumption of the electronic device and the data and operations performed during the execution of the AES algorithm [9]. The attack process performed to find out the key consists of the following steps:

1. The Attack mainly focuses on one of the blocks, either the AddRoundKey block or the SubByte block. The usual preferred point is the output of SubBytes in the first round. The output of AddRoundKey block is also vulnerable but it offers poorer results. In general both the candidates are considered to be attacked first.

2. Let T be the traces associated with the power consumption during the execution of a function that is being captured. Each trace Tj indicates the plain text that is known by the attacker.

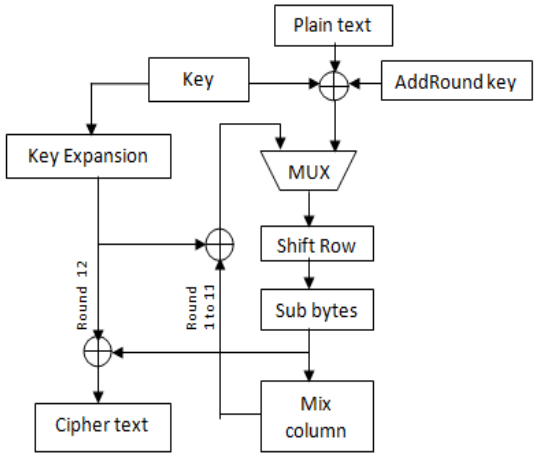

Fig.3 Block diagram for AES-192 bits.

3. In order to simplify the calculation process, the traces are compressed. In our assumption, the average value associated with its power consumption is given by clock cycles. Let us assume that t_a be the point chosen to be attacked. A new vector T_{ta} of length L is generated, which contains all points t_a associated with all compressed traces.

4. The whole process is focused mainly on a particular byte of the function that is chosen in step one. This byte depends only on one byte of the key, which is unknown, along with a byte which is known of the plain text. Only one of the 256 potential keys is correct. For each possible keys a theoretical model of power consumption (hamming-weight) H_k (k=1..256) is created for each element of vector T_{ta}.

5. The correlation coefficient ρ_k (k=1 to 256), between the vector T_{ta} and theoretical model H_k represents the real power consumption, that is calculated.

6. The first byte of the correct cryptographic key corresponds to the highest correlation ρ_k.

7. The process is repeated for each of the 15 remaining key bytes.

III. ALGORITHM STRUTURE TO REVEAL FAKE KEYS

A. Theoretical basis for hiding the true key

The basic idea behind the fake key production is represented in Fig. 4 which is focused on function *SubBytes* that is denoted as SBox. The real key KEY_{REAL} is concealed with a mask $FAKE_{MASK}$ through an exclusive-or operator [9].The result of this operation is the fake key KEY_{FAKE}, which is used for encrypting the plain text that leads to false positive.

It is seen that the system is protected, and if any statistical attack is performed on functions *AddRoundKey* or *SubBytes*then it will reveal the false key KEYFAKE which misleads the attacker. However, some modifications have to be introduced along the encrypting process, the cipher text must be encrypted with the realkey KEY_{REAL}.

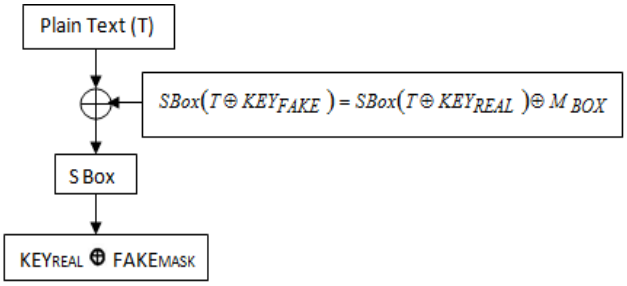

Fig.4 Scheme for encrypting the plain text with a false key.

Note that the output of SBox (the state) can be represented as:

$$SBox(T \oplus KEY_{FAKE}) = SBox(T \oplus KEY_{REAL}) \oplus M_{BOX} \quad (1)$$

where MBOX is a vector of 8-bits length.

Expression (1) allows retrieving the state related to the real key just operating the actual output of Sbox with an exclusive-or. Moreover, M_{BOX} satisfies the following equality:

$$M_{BOX}(j) = SBox(j) \oplus SBox(j \oplus FAKE_{MASK}) \quad (2)$$

Being $j = T \oplus KEY_{FAKE}$. Note that j can take 256 different values. MBOX can be seen as a look-up table of 256 elements, which can be pre-computed only once, at the beginning of the encrypting algorithm:

for (j=0; j<256; j++) {

$M_{BOX}(j) = SBox(j) \oplus SBox(j \oplus FAKE_{MASk})$

}

B. Structure of the modified AES.

The complete architecture proposed in this paper to reveal fake keys is as shown in Fig.5. It can be clearly seen that there are only some small differences between the two models. Further, there are some features that are to be mentioned regarding our proposed system:

1. At any time, the calculations done on the basic functions like *AddRoundKey*, *ShiftRow*, *SubBytes* and *MixColumn*is always based on the false key KEY_{FAKE}. Thus, any statistical attack performed on any of these functions would lead to a false positive (false key).

2. Since the M_{BOX} in our proposed system is pre-computed, so that its execution is very fast and comparable with the function *SubBytes*. Thus, the total execution time is less.

3. Function *MixColumn* has been implemented twice though the second implementation which is connected to the output of M_{BOX}, is not strictly necessary, but we have included it for the success of any attack that was performed in MixColumn.

4. The output of MBOX has been masked with m2 for security purpose. It is a mask which randomly changes for each encrypted plain text. This attempt to prevent a second order attack was carried out by processing the outputs of functions *SubBytes* and MBOX.

It is clear from the block diagram of our proposed model that, the *ShiftRows*is located between *AddRoundKey* and *SubBytes* blocks, so that any attack that has been performed on this function will lead to an identical result similar to the one that we obtained for *AddRoundKey*. Some constrains has been imposed on the attack that is performed on output of *MixColumn* block. Since the *MixColumn* operates on bytes (four bytes) coming from the output of*SubBytes*, key hypotheses of 32 bits should be performed while calculating the correlation coefficient. Usually, attacks based on hypotheses of more than 24 bits are considered unfeasible, in order to overcome this problem, we use plain texts with 3 bytes fixed. The position of these 3 bytes depends on the location of the byte that has to be attacked.

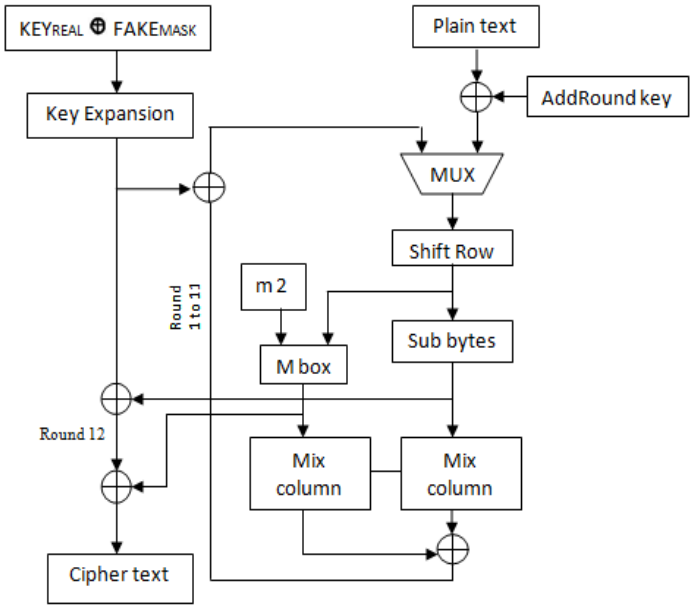

Fig.5 Proposed structure for AES algorithm to reveal fake keys.

For instance, if the target is the byte 4 of the key, then bytes 5, 6 and 7 are fixed. These fixed bytes produced a constant power, which does not affect the correlation coefficient. All cases of simulation have revealed the expected false key KEY_{FAKE}, as the key hypothesis by concealing the true key KEY_{REAL} from the hacker.

IV. SIMULATION RESULTS

A. Simulation results

Several simulations have been performed in ModelSim6.5e software based on the assumptions made on section III and for the sake of simplicity, pre-fixed values has been used for all simulated attacks. On the other hand, mask m2 as in our proposed model is randomly chosen in each encryption process. Fig 6 and 7 shows the simulated output of the encryption and decryption of proposed AES algorithm. The proposed system of obtaining a fake key for the actual text is observed in decryption part of the simulation. A fake key has been obtained for the actual text which acts as the temporary fake key without which the hacker cannot hack the data.

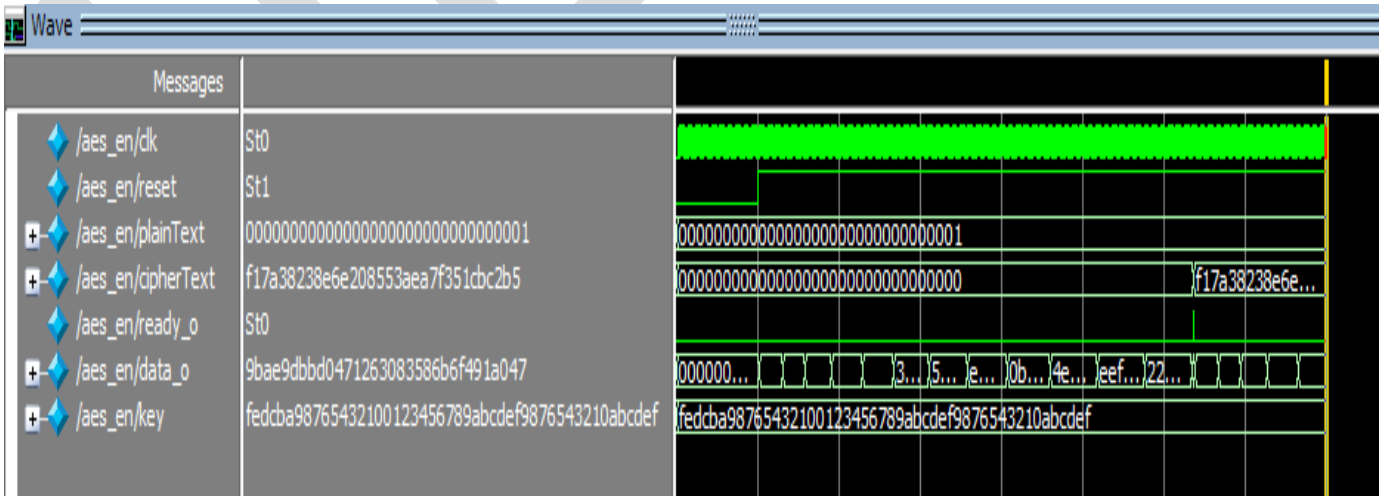

Fig.6 Simulation output of AES-192 Encryption

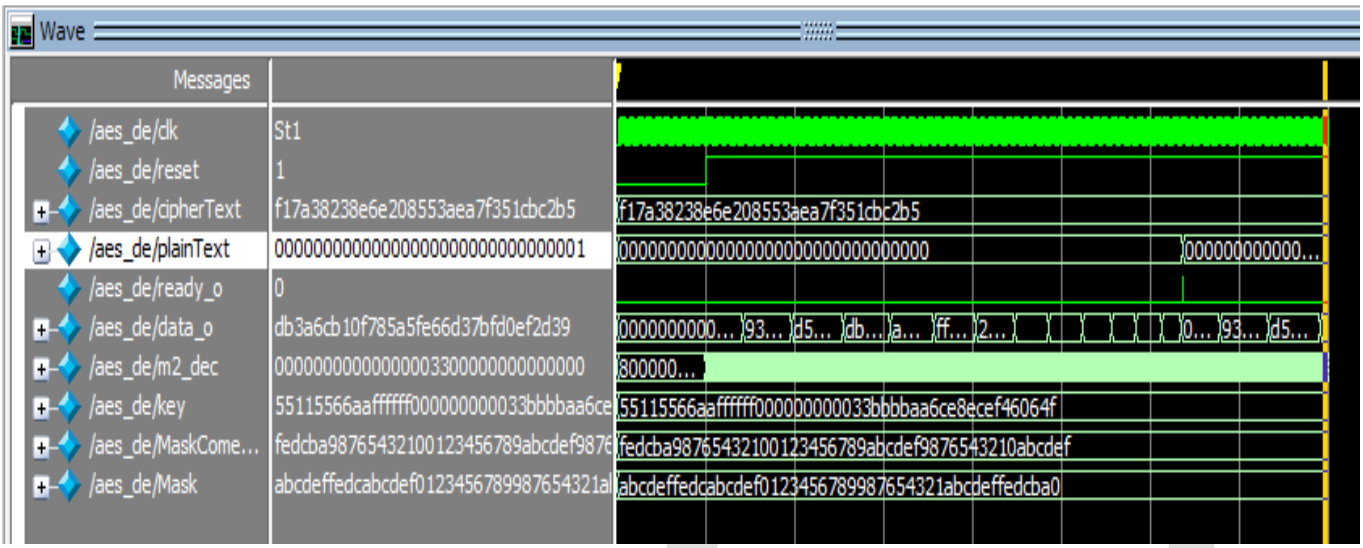

Fig.7 Simulation output of AES-192 Decryption

Fig. 8 and 9, show a comparison of how a brute force attack on both the existing and proposed model is being carried on in MatLabR2014. Here a brute force attack is performed by generating certain number of keys by the attacker and it is being compared with the original key in order to hack the actual data and plotted as a figure of correlation between the generated key and the 192 bit input data

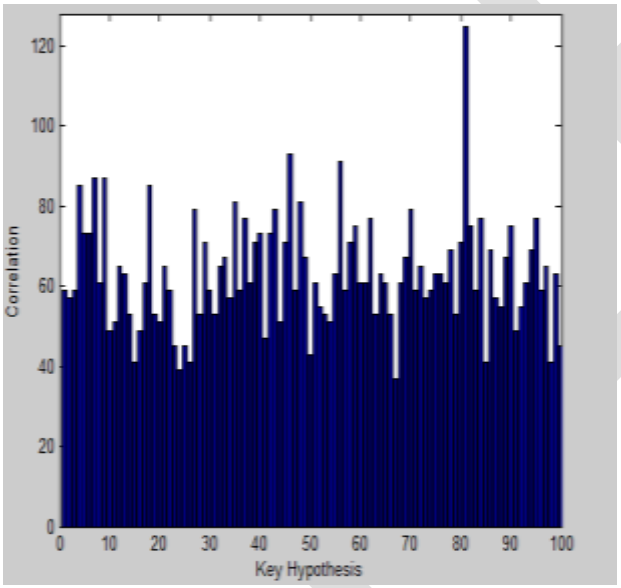

Fig.8 High correlation between actual key and hacker's key.

(Existing Model)

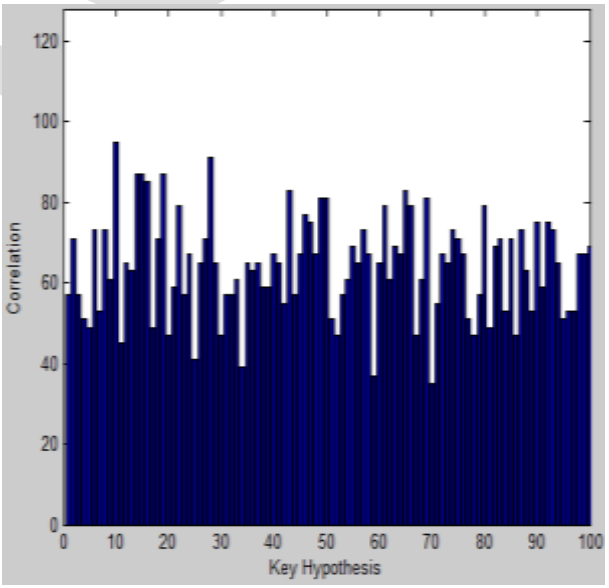

Fig.9 Low correlation between actual key and hacker's key.

(Proposed Model)

Moreover, it is observed that the maximum key is hacked (98%) in existing system without masking scheme indicating that the system is vulnerable and also the time delay was low (0.1157secs). Whereas in our proposed system, when the brute force attack is performed, we find that the correlation between the key and data was found to be minimal (74%) with time delay of 0.6310secs that makes the system reliable and robust when compared to the existing one The experimental results were focused mainly on performing the correlation attacks on functions *AddRoundKey*, *SubBytes* and *MixColumn*.

TABLE I DATA HACK% AND ITS TIME DELAY IN AES-128 AND AES-192[9].

AES Algorithm	Data Hacked	Time Delay
Original AES-128 (with no countermeasures)	98%	0.1045secs
Faked AES-128 (with countermeasures)	84%	0.7321secs
Original AES-192 (with no countermeasures)	98%	0.1157secs
Faked AES-192 (our proposal)	74%	0.6310secs

C. Execution and processing times

Table II shows the execution time of different versions of AES. The original AES-128 and AES-192, with no countermeasures, takes about 16.144 ns and 20.349ns, whereas the protected version using faked AES key is executed in 18.707 ns and 20.341ns. The proposed concept of AES-192 is executed in 20.341ns, including the pre-computation of M_{BOX} and the masking of some intermediate values [9].

TABLE II DELAY TIME FOR DIFFERENT IMPLEMENTATIONS OF AES-128 AND AES-192

AES Algorithm	Delay
Original AES-128 (with no countermeasures)	16.144 ns
Faked AES-128 (with countermeasures)	18.707 ns
Original AES-192 (with no countermeasures)	20.349 ns
Faked AES-192 (our proposal)	20.341 ns

Since the statistical attack is usually performed on the first (1 or 2) or on the last (8, 9 or 10) rounds, the faking countermeasure can be disable during the intermediate rounds (3 to 7) to accelerate the execution time.

TABLE III POWER CONSUMPTION FOR DIFFERENT IMPLEMENTATIONS OF AES-128 AND AES-192

AES Algorithm	Dynamic power (mW)	Static power (mW)
Original AES-128 (with no countermeasures)	457	338
Faked AES-128 (with countermeasures)	295	337
Original AES-192 (with no countermeasures)	347	295
Faked AES-192 (our proposal)	382	295

Table III shows the power consumption (both static and dynamic) for different versions of AES. The original AES-128 and AES-192, with no countermeasures, takes about 457 mW and 347 mW of dynamic power along with 338 mW and 295mW of static power, whereas the protected version using faked AES key for AES-128 and AES-192 consumes about 259 Mw and 382mW of dynamic power along with 337 mW and 295Mw of static power. It is clear from the observation that the proposed scheme has increased power consumption of 10% when compared to convectional model but with the advantage of having increased security of the system [9].

The brute force attack process was programmed in MATLAB R2014a and executed on a laptop clocked at 1.90GHz with a RAM memory of 8.0 GB.

V CONCLUSION

A new countermeasure against side-channel attacks was presented for AES-192. The cryptographic key is protected when a classical statistical attack is performed on any of the functions implemented on the AES algorithm by revealing a false key. The proposal was tested executing the algorithm in ModelsimDE6.5e. Experimental results showed the effectiveness of the proposed method by revealing false keys under brute force attack from hackers using MatLabR2014a Software.

REFERENCES:

[1] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. Advances in Cryptology – Proceedings of Crypto 1999, Lecture Notes in Computer Sicence, vo. 1666, Springer – Verlag, 1999, pp. 338-397.

[2] Darshana Jayasinghe, Roshan Ragel, Jude Angelo Ambrose, Aleksandar Ignjatovic and Sri Parameswaran. Advanced Modes In AES: Are they Safe from Power Analysis based Side Channel Attacks?. Inernational Conference on Computer Design (ICCD),IEEE, 2004, pp.173-180.

[3] Sanchez-Avila,C.;Sanchez-Reillo,R. The Rijndael block cipher (AES Proposal): A comparision with DES. 35th International Conference on Security Technology,IEEE, 2001.pp.229-234.

[4] Kris Tiri and Ingrid Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. Design, Automation and Test in EuropeConference and Exposition, 2004, vol. 1, pp. 246-251.

[5] Rad, A.; ElmSehely,E.; El Hennawy,A.M. Design and Implementation of area optimized AES algorithm on the Reconfigurable FPGA. International conference on Microelectronics (ICM), 2007,pp.35-38.

[6] Sana, P.K; Satyam, M. Alow power secure logic style to Counteract differential power analysis attacks. International symposium on VLSI Design, Automation and Test (VLSI- DAT), 2011, pp.1-4.

[7] Kean Hong BOEY; Hodgers, P.; Yingxi Lu; O'Neill, M.;Woods, R. 17th International Conference on Electronics, Circuits, and Systems(ICECS), IEEE, 2010, pp.1232-1235.

[8] Prouff, E.; Rivain, M,; Bevan, R. Statistical Analysis of SecondOrder Differential Power Analysis. IEEE Transactions onComputers, 2009, pp 799-811.

[9] Savitha.S and Yamuna.S. "ICCCI-International Conference on Computer Communication and Informatics 2016", IEEE Explore.

[10] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger (2011), December 4-8, 'Cryptanalysis of the full AES', the 17th International Conference onTheoryand Application of Cryptologyand InformationSecurity volume 7659, Springer pages 341-352.

[11] Joan Daemen and Vincent Rijmen (2002), 'The Design of Rijndael: AES - The Advanced Encryption Standard', Springer.

[12] National Bureau of Standards, 15 January 1977, Federal Information Processing Standards Publication 46(FIPS PUB 46), Data Encryption Standard (DES), US Department of Commerce.

[13] National Institute of Standards and Technology 2 October 2000, US Department of Commerce, Information related to AES development shall be obtained at http://www.nist.gov/publicaffairs/releases/g00-176.cfm.