Threshold Implementation of a Low-Cost CLEFIA-128 Cipher for Power Analysis Attack Resistance

S. Shanthi Rekha and P. Saravanan(✉)

Department of Electronics and Communication Engineering,
PSG College of Technology, Coimbatore 641 004, India
dpsaravanan@gmail.com

Abstract. Lightweight cryptography aims to satisfy the need for security and privacy in the resource constrained environment like smart cards, RFID and smart edge nodes in Internet of Things (IoT). CLEFIA is one of the ISO/IEC 29191-2 standard lightweight cryptographic algorithm suitable for these applications. Though CLEFIA is proven to be resistant to the cryptanalytic attacks, it is vulnerable to implementation attacks namely Side-Channel Attacks (SCAs). Power Analysis Attacks (PAAs) are the most popular type of SCA and the existing literature has shown successful PAA against CLEFIA. Hence there is a need for strong countermeasure against PAA. The contributions of this work are two-fold: (i) We have proposed a novel 16-bit serial architecture for CLEFIA-128 encryption with a Composite Field Architecture (CFA) based S1 box and Algebraic Normal Form (ANF) based S0 box (ii) A novel Threshold Implementation (TI) with 2-input shares is derived and implemented for the S0, S1 boxes. Thereby, two-shared top level CLEFIA architecture is constructed that shows sufficient first-order PAA resistance when validated using SAKURA-G FPGA board. The PAA categories considered are: (i) Evaluation style – Differential Power Analysis (DPA), Correlation Power Analysis (CPA), Mutual Information Analysis (MIA) with three different power models (ii) Conformance style –Test Vector Leakage Assessment (TVLA) Attack category. This work thereby becomes the first contribution to propose a first-order PAA resistance two-share TI-based CLEFIA implementation with a considerable area compromise, suitable for resource constrained applications.

Keywords: Power Analysis Attack · Threshold Implementation · Low-cost implementation · CLEFIA · Serial architecture

1 Introduction

The recent advancements in Internet of Things (IoT) leads to large number of resource constrained devices to connect to the Internet. To ensure security of these devices, the traditional algorithms are not suitable and hence there is a new class of cryptographic algorithms developed for these applications under the category of lightweight ciphers. Among the different lightweight ciphers developed, Sony Corporation's CLEFIA [1] is standardized by ISO/IEC 29192-2 in 2012 [2]. Further CLEFIA is a recommended

A. Sengupta et al. (Eds.): VDAT 2019, CCIS 1066, pp. 272–285, 2019.
https://doi.org/10.1007/978-981-32-9767-8_24

cipher for lightweight cryptographic technology in e-government applications by the Japanese Cryptographic Research and Evaluation Committee (CRYPTREC) in 2013 [3]. Hence we have implemented CLEFIA in our work.

CLEFIA is a 128-bit block cipher with three different key lengths 128, 192 and 256-bits. Two types of keys namely round keys (RK) and whitening keys (WK) are used for encryption and decryption. The number of rounds of operation varies as 18, 22 and 26 rounds with the key length. The encryption/decryption architecture is based on Generalized Feistel Network (GFN) where a 4-branch Feistel structure is employed with two F-functions F0 and F1. Each of the two F-functions uses two Substitution boxes (S-box) S0 and S1 and two diffusion matrices M0 and M1. The key generation module is also based on the same 4-branch Feistel structure for 128-bit and 8-branch Feistel structure for 192/256-bit key. Decryption is the inverse of encryption where the round keys and whitening keys have to be given in reverse order. More detailed explanation of the round functions and key generation can be found in [1]. While most of the existing works have proposed iterative architectures for CLEFIA-128, the authors of [4] have reported three different 8-bit serialized ASIC implementations for the cipher on 130 nm technology.

The vulnerability of the CLEFIA cipher to Differential Attacks [5] and Fault Attacks [6] has been studied in detail. Side Channel Attacks (SCAs) are one class of attacks that exploits the implementation vulnerabilities of the cryptographic devices to extract the secret key. By monitoring the power consumption/electromagnetic radiations/time taken for execution, the key value can be monitored. These attacks are non-invasive and hence they cannot be detected. Among the different types of side channels, power analysis attacks (PAAs) are vast studied in literature. Authors of [7] have conducted a simulation based Differential Power Analysis (DPA) on the first round of CLEFIA to extract the first round key using a hamming weight model (HWM). The work in [8] demonstrates a Correlation Power Analysis (CPA) and electromagnetic attack on the CLEFIA implementation on 8-bit AVR processor based smart card. The attack is carried out on the last 3 rounds of encryption. Hence there is a strong need for efficient countermeasures to prevent PAA.

The PAA countermeasures can be implemented at different abstraction levels. We aim to implement it at RTL level in-order to implement and validate the same on a SCA standard SAKURA-G board. Two major countermeasures at RTL are Hiding and Masking. The work in [10] has shown first, second and third order masking technique for CLEFIA on FPGA and Intel platform, however there is no evaluation of PAA resistance by the authors. The Masking technique leaks information due to glitches. Threshold Implementation (TI), a variant of masking technique was proposed in [9] to overcome the leakage due to glitches and is proved to be the strongest countermeasure against PAA till date. As per the author's knowledge, there is no work in literature for a TI implementation of CLEFIA. The major contributions of this work are:

- A 16-bit novel serial architecture is proposed for CLEFIA-128 with a Composite Field Architecture (CFA) based S1 box and Algebraic Normal Form (ANF) based S0 box. The proposed architecture has achieved a very low area and a reasonable throughput.

- A novel 2-share based TI implementation is derived for CLEFIA S0 and S1 box for the first time in literature and integrated to derive a two-shared TI based CLEFIA architecture.
- The proposed first-order protected CLEFIA architecture is validated for PAA resistance against (i) Evaluation Style categories – DPA, CPA, Mutual Information Analysis (MIA) and (ii) Conformance Style categories – non-specific Test Vector Leakage Assessment (TVLA). The results show that the implementation is resistant against first-order PAAs.

The paper is organized as follows: Sect. 2 describes the novelty in CLEFIA architecture along with the result comparison. Section 3 explains in detail on the proposed TI implementation derivation and Sect. 4 presents the results of PAA evaluation. Section 5 presents the conclusion followed by references.

2 Proposed CLEFIA Architecture

We have proposed a serial CLEFIA architecture with a 16-bit data-path to perform encryption/decryption as shown in Fig. 1. The synthesis results are performed with Semi-Conductor Laboratory (SCL) [8] 180 nm technology using Cadence Encounter RTL Compiler. For comparison purposes, the area is reported in Gate Equivalence (GE) which is obtained by dividing the area reported in μm^2 by the area of the lowest driving strength 2-input NAND gate of that corresponding technology library.

The data-path is formed by using area efficient S0/S1 boxes as explained in Sect. 2.1, followed by M0/M1 matrix realizations using x2, x4 and x8 Galois Field (GF) multipliers. The register files R0–R7 are of 16-bit size and each line carries 16-bit data except the ones explicitly shown as 8-bits. The addition of R8 register file in our proposed serial architecture has reduced the critical path delay from 5.1 ns to 3.5 ns. Further the '16' 2x1 MUXes required for the product addition in M0/M1 matrix Multipliers are reduced to '2' MUXes by placing them before the 'R8' register. These optimization techniques along with the efficient realizations of the individual blocks to obtain a 16-bit CLEFIA data-path is one of our contribution.

2.1 ANF Based CLEFIA S0 Box

CLEFIA S0 box construction is based on 4 random S-boxes namely SS0, SS1, SS2 and SS3 which are further combined using GF(2^4) multiplier. In this work, ANF based implementation is chosen for the S0 box realization. ANF representation of a Boolean function is useful to evaluate the algebraic degree and the linearity property of any function. ANF expressions are mainly useful to find shares of the Boolean function, when we adopt the countermeasure against SCA namely TI. The ANF implementation of S0 box requires only 115 GE, hence we have adopted the same for our serial architecture.

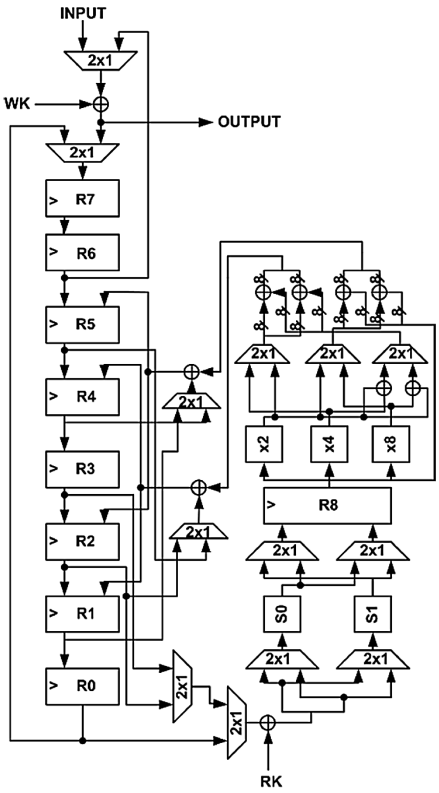

Fig. 1. Proposed 16-bit serial data-path of CLEFIA architecture

2.2 CFA Based CLEFIA S1 Box

CLEFIA S1 box construction is based on GF(2^8) inversion using the polynomial $z^8 + z^4 + z^3 + z^2 + 1$ [2]. The CFA based realization of S-boxes are proved to achieve the lowest area in literature. The core of the CFA computation is the multiplicative inverse block preceded and followed by the affine transformation f and g respectively as shown in Fig. 2. Gate-level implementations of the GF(2^4) arithmetic based blocks namely multiplier, squarer, constant multiplier, inverter, isomorphic and inverse isomorphic mapping combined with affine f and g are adopted in this work. The CFA based S1 box architecture consumes an area of about 207 GE which is 50% less compared to the LUT approach. Hence we have chosen this for the top level serial architecture.

The proposed serial architecture after integration of the sub-blocks occupies 1345 GE and consumes a power of 1.3 mW. The performance comparison of the proposed serial CLEFIA architecture with the existing serial architecture [4] is shown in Table 1. It can be seen that, there is about 19% decrease in total area and a two-fold improvement in throughput calculated at 100 kHz. For comparison with [4], we have chosen the operating frequency of 100 kHz. The area decrease of the proposed

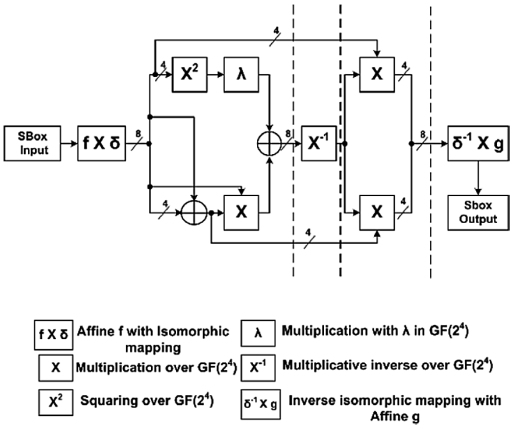

Fig. 2. Proposed CFA architecture of CLEFIA S1 box

architecture is attributed to low cost implementation of S0 and S1 boxes. The decrease in number of clock cycles which has resulted in throughput improvement is because of the optimal selection of data-path size of 16 bits against the 8-bit size in [4]. Each round computation can be completed in 7 clock cycles, an additional 8 clock cycles is required for input loading and output retrieval, thereby a total of 142 clock cycles for encryption/decryption. Also, the proposed architecture can be used for both encryption and decryption.

The proposed serial CLEFIA architecture occupies the lowest area and has a very low power consumption and hence suitable for resource constrained applications like edge nodes in IoT devices, RFID Tags, smart cards etc.

3 Proposed CLEFIA Threshold Implementation

The proposed serial architecture has comparatively less algorithmic noise that arises from parallel computation of data and hence has less resistance to PAA. This motivates us to derive a low-cost RTL countermeasure for the proposed serial CLEFIA architecture which is explained in the subsequent section.

3.1 Threshold Implementation of CLEFIA

The objective of the countermeasures is to nullify the effect of data that the circuit processes on the power consumption. At the RTL level, the most efficient technique proposed so far till date is Threshold Implementation (TI), originally proposed by Nikova et al. in 2006 [9]. It involves splitting of the data into two or more shares and the computation data-path of each share runs in parallel. There are three basic requirements that any TI implementation should satisfy: Correctness, Non-completeness and Uniformity. The key features and advantages of TI techniques are (i) The intermediate results do not leak any information about the secret value since

each share is independent of at least one input (ii) Even in the presence of glitches, sufficient first-order resistance is obtained. (iii) The technique is compatible with the standard RTL Design flow.

In general for a 'd'-th order security of any arbitrary function with algebraic degree 't', the TI scheme uses a minimum of (td + 1) shares. The work in [11] is to further reduce the number of shares from (td + 1) to (d + 1) provided that the input shares are independent. Hence the output share is independent of at least one share of input variable. In-order to satisfy the non-completeness property, the design has to be pipelined to sufficient number of stages. To ensure uniformity in all these cases, sufficient bits of randomness is introduced at each pipelined intermediate computation stage. The work in [12] has realized a 3-stage pipelined AES S-box with 2 shares with each stage requiring about 12, 24 and 28 bits of randomness. The authors claim to have achieved less area compared to the other AES TI implementations.

Based on this idea, ours is the first work to derive a TI implementation for a low-cost CLEFIA architecture. The TI implementation of the proposed CFA S1 box and ANF S0 boxes are derived and using the same, the two-share top level CLEFIA architecture is implemented. In this work, the shares of the inputs a, b are denoted as *{a0, a1} {b0, b1}*; the shares of the output f[i], i = 0, 1, 2, 3 are denoted as *{f3[i], f2[i], f1[i], f0[i]}* respectively.

TI Implementation of CLEFIA S1 box: The CFA Architecture of S1 box is shown in Sect. 2.2. The CFA architecture consists of linear blocks namely affine transformation f with isomorphic mapping, affine transformation g with inverse isomorphic mapping, square-scaler, addition blocks which are replicated directly for processing the two-shares. The non-linear blocks are the GF(2^4) multiplier and inverter. Based on the work in [11], for the first-order security (d = 1), number of required input shares is d + 1=2, provided that the input shares are independent and hence the number of output shares is $(d + 1)^t = 2^t$, *t* is the algebraic degree of the function. It is observed that the algebraic degree of the GF(2^4) multiplier is 2 and GF(2^4) inverter is 3. Hence the TI multiplier implementation takes 2-input and produces 4-output shares as shown in (1), inverter implementation takes 2-input and produces 8-output shares. The 4-output shares of multiplier are further reduced to two by xoring with sufficient bits of randomness. Similarly the 8-output shares of inverter are reduced to two. Also, the number of pipeline stages is 3 similar to [12]. Due to space limitations, only the multiplier sharing expressions are given in (1). To satisfy the non-uniformity property, the 3-pipelining stages require 12, 28 and 24 bits of randomness each. The TI implementation of CFA S1 box occupies 1085 GE and consumes 6.2 mW power.

$$
\begin{aligned}
f[3] &= (a[3] * b[3]) \oplus (a[3] * b[0]) \oplus (a[0] * b[3]) \oplus (a[2] * b[1]) \oplus (a[1] * b[2]) \\
f[2] &= (a[3] * b[3]) \oplus (a[3] * b[2]) \oplus (a[2] * b[3]) \oplus (a[2] * b[0]) \oplus (a[0] * b[2]) \oplus (a[1] * b[1]) \\
f[1] &= (a[3] * b[2]) \oplus (a[2] * b[3]) \oplus (a[3] * b[1]) \oplus (a[1] * b[3]) \oplus (a[2] * b[2]) \oplus (a[0] * b[1]) \oplus (a[1] * b[0]) \\
f[0] &= (a[3] * b[1]) \oplus (a[1] * b[3]) \oplus (a[2] * b[2]) \oplus (a[0] * b[0])
\end{aligned}
$$

$$f0[3] = (a0[3] * b0[3]) \oplus (a0[3] * b0[0]) \oplus (a0[0] * b0[3]) \oplus (a0[2] * b0[1]) \oplus (a0[1] * b0[2])$$
$$f1[3] = (a0[3] * b1[3]) \oplus (a0[3] * b1[0]) \oplus (a0[0] * b1[3]) \oplus (a0[2] * b1[1]) \oplus (a0[1] * b1[2])$$
$$f2[3] = (a1[3] * b0[3]) \oplus (a1[3] * b0[0]) \oplus (a1[0] * b0[3]) \oplus (a1[2] * b0[1]) \oplus (a1[1] * b0[2])$$
$$f3[3] = (a1[3] * b1[3]) \oplus (a1[3] * b1[0]) \oplus (a1[0] * b1[3]) \oplus (a1[2] * b1[1]) \oplus (a1[1] * b1[2])$$

$$f0[2] = (a0[3] * b0[3]) \oplus (a0[3] * b0[2]) \oplus (a0[2] * b0[3]) \oplus (a0[2] * b0[0]) \oplus (a0[0] * b0[2]) \oplus (a0[1] * b0[1])$$
$$f1[2] = (a0[3] * b1[3]) \oplus (a0[3] * b1[2]) \oplus (a0[2] * b1[3]) \oplus (a0[2] * b1[0]) \oplus (a0[0] * b1[2]) \oplus (a0[1] * b1[1])$$
$$f2[2] = (a1[3] * b0[3]) \oplus (a1[3] * b0[2]) \oplus (a1[2] * b0[3]) \oplus (a1[2] * b0[0]) \oplus (a1[0] * b0[2]) \oplus (a1[1] * b0[1])$$
$$f3[2] = (a1[3] * b1[3]) \oplus (a1[3] * b1[2]) \oplus (a1[2] * b1[3]) \oplus (a1[2] * b1[0]) \oplus (a1[0] * b1[2]) \oplus (a1[1] * b1[1])$$

$$f0[1] = (a0[3] * b0[2]) \oplus (a0[2] * b0[3]) \oplus (a0[3] * b0[1]) \oplus (a0[1] * b0[3]) \oplus (a0[2] * b0[2]) \oplus (a0[0] * b0[1]) \oplus (a0[1] * b0[0])$$
$$f1[1] = (a0[3] * b1[2]) \oplus (a0[2] * b1[3]) \oplus (a0[3] * b1[1]) \oplus (a0[1] * b1[3]) \oplus (a0[2] * b1[2]) \oplus (a0[0] * b1[1]) \oplus (a0[1] * b1[0])$$
$$f2[1] = (a1[3] * b0[2]) \oplus (a1[2] * b0[3]) \oplus (a1[3] * b0[1]) \oplus (a1[1] * b0[3]) \oplus (a1[2] * b0[2]) \oplus (a1[0] * b0[1]) \oplus (a1[1] * b0[0])$$
$$f3[1] = (a1[3] * b1[2]) \oplus (a1[2] * b1[3]) \oplus (a1[3] * b1[1]) \oplus (a1[1] * b1[3]) \oplus (a1[2] * b1[2]) \oplus (a1[0] * b1[1]) \oplus (a1[1] * b1[0])$$

$$\begin{aligned} f0[0] &= (a0[3] * b0[1]) \oplus (a0[1] * b0[3]) \oplus (a0[2] * b0[2]) \oplus (a0[0] * b0[0]) \\ f1[0] &= (a0[3] * b1[1]) \oplus (a0[1] * b1[3]) \oplus (a0[2] * b1[2]) \oplus (a0[0] * b1[0]) \\ f2[0] &= (a1[3] * b0[1]) \oplus (a1[1] * b0[3]) \oplus (a1[2] * b0[2]) \oplus (a1[0] * b0[0]) \\ f3[0] &= (a1[3] * b1[1]) \oplus (a1[1] * b1[3]) \oplus (a1[2] * b1[2]) \oplus (a1[0] * b1[0]) \end{aligned} \tag{1}$$

TI Implementation of CLEFIA S0 box: The ANF expressions of the CLEFIA S0 box has an algebraic degree of 3. Hence the SS0, SS1, SS2 and SS3 sub-blocks are shared with 2-input and 8-outputs (further reduced to 2) each and are combined using GF(2) multipliers to arrive at its TI implementation. Such an implementation has occupied about 581 GE with a power consumption value of 5.2 mW. Due to page limitations, ANF expressions of only SS0 and its corresponding two shared TI implementation expressions are shown in (2).

$$f0 = 1 \oplus a[0] \oplus (a[3] * a[2]) \oplus (a[3] * a[1]) \oplus (a[2] * a[1]) \oplus (a[1] * a[0]) \oplus (a[3] * a[2] * a[1]) \oplus (a[2] * a[1] * a[0])$$
$$f1 = 1 \oplus a[3] \oplus a[2] \oplus (a[3] * a[1]) \oplus (a[2] * a[0]) \oplus (a[1] * a[0]) \oplus (a[3] * a[2] * a[1]) \oplus (a[2] * a[1] * a[0])$$
$$f2 = 1 \oplus a[2] \oplus a[1] \oplus (a[3] * a[0]) \oplus (a[2] * a[0]) \oplus (a[1] * a[0]) \oplus (a[3] * a[2] * a[1])$$
$$f3 = a[3] \oplus (a[3] * a[1]) \oplus (a[2] * a[0]) \oplus (a[3] * a[2] * a[1]) \oplus (a[3] * a[2] * a[0])$$

$$f0[0] = 1 \oplus a0[0] \oplus (a0[3] * a0[2]) \oplus (a0[3] * a0[1]) \oplus (a0[2] * a0[1]) \oplus (a0[1] * a0[0]) \oplus (a0[3] * a0[2] * a0[1]) \oplus (a0[2] * a0[1] * a0[0])$$
$$f1[0] = a1[0] \oplus (a0[3] * a1[2]) \oplus (a0[3] * a1[1]) \oplus (a1[2] * a1[1]) \oplus (a1[1] * a1[0]) \oplus (a0[3] * a1[2] * a1[1]) \oplus (a1[2] * a1[1] * a1[0])$$
$$f2[0] = (a1[3] * a0[2]) \oplus (a1[3] * a1[1]) \oplus (a0[2] * a1[1]) \oplus (a1[1] * a0[0]) \oplus (a1[3] * a0[2] * a1[1]) \oplus (a0[2] * a1[1] * a0[0])$$
$$f3[0] = (a1[3] * a1[2]) \oplus (a1[3] * a0[1]) \oplus (a1[2] * a0[1]) \oplus (a0[1] * a1[0]) \oplus (a1[3] * a1[2] * a0[1]) \oplus (a1[2] * a0[1] * a1[0])$$
$$f4[0] = (a0[3] * a0[2] * a1[1]) \oplus (a0[2] * a1[1] * a1[0])$$
$$f5[0] = (a0[3] * a1[2] * a0[1]) \oplus (a1[2] * a0[1] * a0[0])$$
$$f6[0] = (a1[3] * a0[2] * a0[1]) \oplus (a0[2] * a0[1] * a1[0])$$
$$f7[0] = (a1[3] * a1[2] * a1[1]) \oplus (a1[2] * a1[1] * a0[0])$$

$$
\begin{aligned}
f0[1] &= 1 \oplus a0[3] \oplus a0[2] \oplus (a0[3] * a0[1]) \oplus (a0[2] * a0[0]) \oplus (a0[1] * a0[0]) \oplus (a0[3] * a0[2] * a0[1]) \oplus (a0[2] * a0[1] * a0[0]) \\
f1[1] &= a1[3] \oplus a1[2] \oplus (a1[3] * a0[1]) \oplus (a1[2] * a1[0]) \oplus (a0[1] * a1[0]) \oplus (a1[3] * a1[2] * a0[1]) \oplus (a1[2] * a0[1] * a1[0]) \\
f2[1] &= (a0[3] * a1[1]) \oplus (a0[2] * a1[0]) \oplus (a1[1] * a1[0]) \oplus (a0[3] * a0[2] * a1[1]) \oplus (a0[2] * a1[1] * a1[0]) \\
f3[1] &= (a1[3] * a1[1]) \oplus (a1[2] * a0[0]) \oplus (a[1] * a0[0]) \oplus (a1[3] * a1[2] * a1[1]) \oplus (a1[2] * a1[1] * a0[0]) \\
f4[1] &= (a0[3] * a1[2] * a0[1]) \oplus (a1[2] * a0[1] * a0[0]) \\
f5[1] &= (a1[3] * a0[2] * a0[1]) \oplus (a0[2] * a0[1] * a1[0]) \\
f6[1] &= (a0[3] * a1[2] * a1[1]) \oplus (a1[2] * a1[1] * a1[0]) \\
f7[1] &= (a1[3] * a0[2] * a1[1]) \oplus (a0[2] * a1[1] * a0[0])
\end{aligned}
$$

$$
\begin{aligned}
f0[2] &= 1 \oplus a0[2] \oplus a0[1] \oplus (a0[3] * a0[0]) \oplus (a0[2] * a0[0]) \oplus (a0[1] * a0[0]) \oplus (a0[3] * a0[2] * a0[1]) \\
f1[2] &= a1[2] \oplus a1[1] \oplus (a0[3] * a1[0]) \oplus (a1[2] * a1[0]) \oplus (a1[1] * a1[0]) \oplus (a0[3] * a1[2] * a1[1]) \\
f2[2] &= (a1[3] * a0[0]) \oplus (a1[2] * a0[0]) \oplus (a1[1] * a0[0]) \oplus (a1[3] * a1[2] * a1[1]) \\
f3[2] &= (a1[3] * a1[0]) \oplus (a0[2] * a1[0]) \oplus (a0[1] * a1[0]) \oplus (a1[3] * a0[2] * a0[1]) \\
f4[2] &= (a0[3] * a0[2] * a1[1]) \\
f5[2] &= (a0[3] * a1[2] * a0[1]) \\
f6[2] &= (a1[3] * a0[2] * a1[1]) \\
f7[2] &= (a1[3] * a1[2] * a0[1])
\end{aligned}
$$

$$
\begin{aligned}
f0[3] &= a0[3] \oplus (a0[3] * a0[1]) \oplus (a0[2] * a0[0]) \oplus (a0[3] * a0[2] * a0[1]) \oplus (a0[3] * a0[2] * a0[0]) \\
f1[3] &= a1[3] \oplus (a1[3] * a0[1]) \oplus (a0[2] * a1[0]) \oplus (a1[3] * a0[2] * a0[1]) \oplus (a1[3] * a0[2] * a1[0]) \\
f2[3] &= (a0[3] * a1[1]) \oplus (a1[2] * a0[0]) \oplus (a0[3] * a1[2] * a1[1]) \oplus (a0[3] * a1[2] * a0[0]) \\
f3[3] &= (a1[3] * a1[1]) \oplus (a1[2] * a1[0]) \oplus (a1[3] * a1[2] * a1[1]) \oplus (a1[3] * a1[2] * a1[0]) \\
f4[3] &= (a0[3] * a0[2] * a1[1]) \oplus (a0[3] * a0[2] * a1[0]) \\
f5[3] &= (a0[3] * a1[2] * a0[1]) \oplus (a0[3] * a1[2] * a1[0]) \\
f6[3] &= (a1[3] * a0[2] * a1[1]) \oplus (a1[3] * a0[2] * a0[0]) \\
f7[3] &= (a1[3] * a1[2] * a0[1]) \oplus (a1[3] * a1[2] * a0[0])
\end{aligned}
\tag{2}
$$

The two-share S0 and S1 boxes are then integrated to form the top level two-share architecture by replicating the linear blocks namely key XORs, registers, x1, x4, x8 multipliers and multiplexers twice for processing the two input shares. The two shares of the plain text are derived by using a LFSR-based Pseudo-Random Number Generator (PRNG) block which generates a 16-bit random number. Thereby the top-level serial based two-shared CLEFIA Architecture occupies around 3955 GE which is about three times more than the unprotected implementation with a power consumption of 30 mW. The throughput decreases by 20% because of the extra clock cycles required due to the pipelined S0 and S1 boxes. Based on our survey, since there is no TI work on CLEFIA in the existing literature, we compare our results with that of AES cipher. A recent low-cost AES TI with similar number of shares have reported an area of 6053 GE on a byte serial architecture [12]. However the proposed lightweight CLEFIA cipher with an efficient TI implementation has 34% lesser area, making it still suitable for resource constrained applications. Since the power consumption is technology dependent, we do not attempt to compare the same with other works.

Table 1. Synthesis results comparison

Reference	Area (in GE)	No. of clock cycles	Throughput @ 100 kHz(kbps)	Efficiency
Existing serial [4]	1664	328	39	0.023
Our serial unprotected	**1345**	**142**	**90**	**0.067**
Our serial protected	**3955**	**178**	**71.91**	**0.018**

4 PAA Resistance Analysis

PAAs are Known Plain Text/Cipher Text attacks that recover the secret key byte-by-byte. The first part of this section explains the setup developed to perform the PAA while the second part describes the results obtained for protected and unprotected implementations.

4.1 PAA Setup

- Threat model: It is assumed that the attacker has physical access to the cryptographic device implementing the encryption algorithm and can control the data inputs. The attacker can also observe the outputs of the performed operation, without the knowledge of the secret key.
- Attack Data-path: The selection function in our work is the output of the 8-bit S-boxes S0/S1 of first round since S-boxes are responsible to induce the confusion property of the cipher. The data-path computes and stores one S-box computation per clock cycle.
- Predictions Phase: For 'm' plain-text values, the possible key values for an 8-bit attack path is $k_s(i) = 0$ to 255. The data-path is evaluated using MATLAB code and the S-box output values are obtained. The size of the S-box output matrix is m x 256. The DPA attack work on bit models (any bit of the S-box) and CPA/MIA can be performed using HWM or Hamming Distance Model (HDM) or Signed Bit Model (SBM).
- Measurement Phase: The implementation of the attack data-path is done on SCA Standard Evaluation Board (SASEBO) namely SAKURA-G which is embedded with two FPGAs: (i) The main FPGA namely Cryptographic FPGA – Spartan 6 xc6slx75 and (ii) The secondary FPGA namely Control FPGA – Spartan 6 xc6slx9. The amplified and de-noised supply current of the design implemented in Main FPGA is tapped through SMA coaxial cables. The current traces are captured using KeySight Mixed Signal Oscilloscope MSO6014A. The oscilloscope captures the trigger signal and the current trace from the FPGA board.
- Pre-processing and Attack Phase: The efficient pre-processing phase of our work consists of the following features: (i) The measured current samples contain very

little noise since all the capacitors on Vcore line are removed and voltage drop (power trace) caused by the 1-ohm shunt resistor inserted between the cryptographic FPGA and the Vcore line is captured. (ii) The measured current samples contain a sufficient current level for the attack since they are obtained from the in-built amplifier on the board with a bandwidth of 360 MHz and a gain of +20 dB. (iii) The required features of the measured current samples are extracted using Python. The pre-processed samples are then re-arranged into a trace matrix *m x n*. The pre-processed traces are then analyzed using DPA-CPA/MIA/TVLA attack scripts in MATLAB.

4.2 Attack Results

With our developed attack setup, we have performed two styles of PAA: (i) Evaluation style attacks- which involves forming a theoretical model and evaluating the implementation against different attack strategies. The analysis types like DPA [13], CPA [14] and MIA [15] fall under this category. DPA uses difference of means (DoM) method to predict the key while CPA uses a statistical distinguisher namely Pearson's Correlation Coefficient (CC) to correlate between the measurements obtained and predictions made. The idea of using information theory metrics to quantify the dependence between the leakage and power consumption model was proposed by Benedikt et al., as MIA in [15]. The advantage of this method is that it relaxes the strict requirement of linear dependency between the measurements and predictions as in CPA and exploits comparatively more information than DPA. Hence the MIA based attacks are independent of the attacking platform and the variables need not be strictly linear. In our work, the Mutual Information (MI) between the random variable X (power consumption) and the leakage L (predictions) is calculated in bits using MATLAB code. For each secret key value, MI function processes the inputs column by column and returns an 'm x n' matrix, where 'm' corresponds to the number of traces and 'n' corresponds to the sampling points. The maximum value of MI is subsequently recorded, the index of which returns the key. Our work reports a metric named Minimum Time to Disclosure (MTD) that describes the cross-over point of the correct Correlation Coefficient (CC) with the CC curves of the other key guesses. We have used this metric to quantify the CPA and MIA attacks in our work. (ii) Conformance style attacks- which involves evaluating the leakages independently, to determine if there is a SCA vulnerability, rather than estimating the correct key. The very popular TVLA [16] falls under this category which performs t-tests using the statistical mean and variance parameters. Based on the literature, acceptable t-value across all the sampling points is $\leq \pm 4.5$. If t-value exceeds the threshold, the implementation is considered to be vulnerable to PAA. This methodology does not determine the exact secret key value or the number of traces required for successful key retrieval. In certain applications it is required to strongly determine the implementation vulnerability of a system and this methodology provides its usefulness in those scenarios.

Table 2 shows the metrics comparison of protected and unprotected implementations of a few keys. While the un-protected implementations have shown successful key recoveries with less than 768 traces for all the attack categories, the protected implementation does not reveal the secret key even with 100,000 traces for first-order attacks. Hence sufficient PAA protection is achieved using the proposed two-share CLEFIA architecture. Figures 3, 4 and 5 shows the attack results of CPA and TVLA.

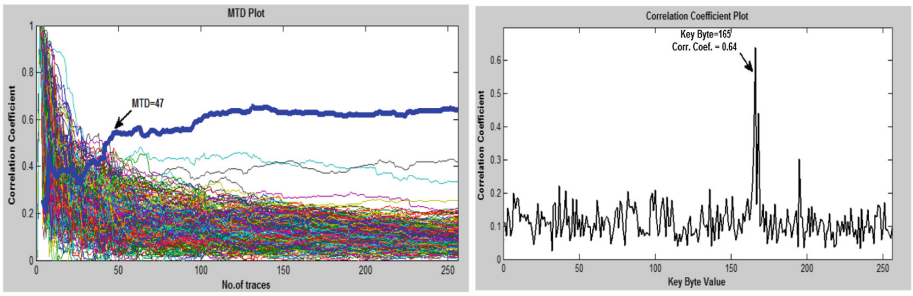

Fig. 3. (a) CPA-HWM CC plot (b) MTD plot for key byte = 165 on unprotected CLEFIA S0 box

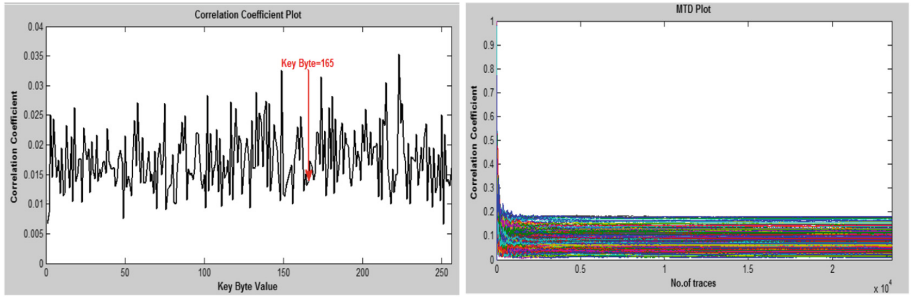

Fig. 4. (a) CPA-HWM CC plot (b) MTD plot for key byte = 165 on protected CLEFIA S0 box

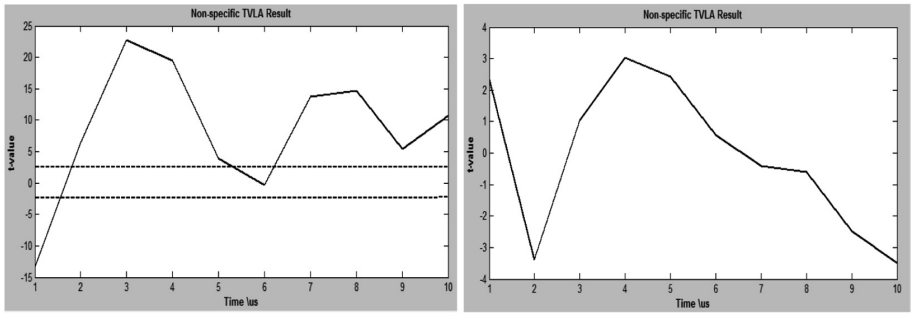

Fig. 5. Non-specific TVLA results for (a) un-protected (b) protected implementation

Table 2. PAA resistance analysis

Cipher	Attack category	Key-byte value (in Hex)	Metrics achieved	No. of traces used
Unprotected CLEFIA	DPA	90(S0 box)	Max. diff = 0.6	256
	CPA-HWM	90(S0 box)	CC = 0.6, MTD = 37	256
	CPA-HDM	165(S0 box)	CC = 0.35, MTD = 175	256
	CPA-SBM	90(S1 box)	CC = 0.26, MTD=182	256
	MIA-HWM	90(S0 box)	MI = 0.78, MTD = 240	768
	MIA-HDM	165(S0 box)	MI = 0.6, MTD = 540	768
	MIA-SBM	165(S0 box)	MI = 0.35, MTD = 225	768
	TVLA	90(S1 box)	t-value > −4.5	512
Protected CLEFIA	Evaluation style (First-Order)	90(S0/S1 box) 165(S0 box)	Unsuccessful	100,000
	Conformance style (First-order)	90(S0/S1 box) 165(S0 box)	t-value < 4.5	100,000

5 Conclusion

CLEFIA is an ISO/IEC standard lightweight cryptographic algorithm suitable for resource constrained applications. We have proposed a 16-bit novel serial CLEFIA architecture with a low-cost CFA based S1 box and ANF based S0 box. The proposed serial architecture has achieved a very low area of 1345 GE with a power consumption of 1.3 mW. Since PAA is shown to be a stronger threat to the cryptographic implementations, we have developed a novel two-share TI implementation for the proposed CLEFIA architecture to mitigate the same. The proposed TI implementation occupies an area of 3955 GE, which is 34% smaller than a similar AES TI implementation. We have briefed the PAA setup created using SAKURA-G board and performed the Evaluation and Conformance style attacks. While the unprotected implementation is easily attacked with less than 768 traces, the protected implementation shows resistance to PAA using 100,000 traces.

The first-order PAA resistant CLEFIA architecture is suitable for resource constrained applications. The higher order PAA resistance of the proposed TI implementation will be analyzed in future.

Acknowledgement. This work has been done from the grant received from Visvesvaraya PhD Scheme of Ministry of Electronics and Information Technology, Government of India, being implemented by Digital India Corporation. This work was also supported by Special Manpower Development Programme for Chips to System Design (SMDP-C2SD) project sponsored by the Department of Electronics and Information Technology (DeitY), Government of India. The

authors would like to thank the contribution from the following people: Vidhya D, Aravindh M, Kirubhas Shankar R K, Lakshmanan G, Mahalakshmi S, Prashanth S and Rajesh Srivatsav S, Department of ECE, PSG College of Technology for their support in implementation of the proposed work.

References

1. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit blockcipher CLEFIA (extended abstract). In: biryukov, a (ed.) FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74619-5_12
2. ISO, ISO/IEC 29192-2:2012: Information Technology - Security Techniques - Lightweight Cryptography - Part 2: Block Ciphers (2012). http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=56552
3. CRYPTREC: Cryptographic Technology Guideline. http://www.cryptrec.go.jp/report/cryptrec-rp-2000-2017.pdf
4. Akishita, T., Hiwatari, H.: Very compact hardware implementations of the blockcipher CLEFIA. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 278–292. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28496-0_17
5. Boura, C., Naya-Plasencia, M., Suder, V.: Scrutinizing and improving impossible differential attacks: applications to CLEFIA, camellia, LBlock and Simon. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 179–199. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45611-8_10
6. Chester, R., Mukhopadhyay, D.: Differential Cache Trace Attack against CLEFIA. IACR Cryptology ePrint Archive, 12, (2010)
7. Bai, X., Lu, H., Wang, Y., Xu, Y.: Differential power analysis attack on CLEFIA block cipher. In: IEEE International Conference on Computational Intelligence and Software Engineering, pp. 1–4 (2009)
8. Kim, Y., Ahn, J., Choi, H.: Power and electromagnetic analysis attack on a smart card implementation of CLEFIA. In: International Conference on Security and Management (SAM), p. 1 (2013)
9. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 529–545. Springer, Heidelberg (2006). https://doi.org/10.1007/11935308_38
10. Baihan, A., Duggirala, P.S., Baihan, M.: A high-order masking approach for CLEFIA implementation on FPGA and Intel. In: Proceedings of the International Conference on Security and Management (SAM), pp. 79–85 (2017)
11. Reparaz, O., Bilgin, B., Nikova, S., Gierlichs, B., Verbauwhede, I.: Consolidating masking schemes. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 764–783. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_37
12. Ueno, R., Homma, N., Aoki, T.: Toward more efficient DPA-resistant AES hardware architecture based on threshold implementation. In: Guilley, S. (ed.) COSADE 2017. LNCS, vol. 10348, pp. 50–64. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64647-3_4
13. Kocher, P., Jaffe, J., Jun, B., Rohatgi, P.: Introduction to differential power analysis. J. Cryptographic Eng. **1**, 5–27 (2011)
14. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_2

15. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual information analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85053-3_27
16. Roy, D.B., Bhasin, S., Patranabis, S., Mukhopadhyay, D., Guilley, S.: What Lies Ahead: Extending TVLA Testing Methodology Towards Success Rate. IACR Cryptology ePrint Archive, 1152 (2016)