Journal of Hardware and Systems Security (2019) 3:103–131
https://doi.org/10.1007/s41635-018-0049-y

Lightweight Design-for-Security Strategies for Combined Countermeasures Against Side Channel and Fault Analysis in IoT Applications

Sikhar Patranabis[1] · **Debapriya Basu Roy**[1] · **Anirban Chakraborty**[1] · **Naveen Nagar**[2] · **Astikey Singh**[2] · **Debdeep Mukhopadhyay**[1] · **Santosh Ghosh**[3]

Received: 15 March 2018 / Accepted: 17 September 2018 / Published online: 26 September 2018

Abstract
The Internet-of-Things today gives rise to a number of applications that require lightweight cryptographic primitives, such as block ciphers for secure and efficient computation using very little resources. This paper addresses the open problem of design-for-security methodologies for constructing such lightweight block ciphers with combined protection against both side channel and fault attacks. We propose novel design strategies that, unlike existing methodologies, are equipped with target-specific design choices. Our first proposal is the incorporation of lightweight linear layers that combine good diffusion properties with fault attack resistance via fault space transformation. Our second proposal is to make S-Box choices using a new metric called the modified transparency order, so as to facilitate a lightweight masking strategy where the mask is only periodically refreshed. Our third and final proposal is to implement a cipher-dependent multi-round shuffling technique that is lightweight and affords greater security than the standard shuffling schemes in the literature. Each of our propositions are assembled into one single construction for a PRESENT-like block cipher, that consumes 15% less look-up tables on a Xilinx xc5vlx50 FPGA than all existing threshold implementations of PRESENT, and provides good security guarantees against both fault and side-channel attacks. In particular, it resists both classical and biased fault attacks, and does not exceed the safety threshold against side-channel attacks over 50,000 power traces, collected on a SASEBO GII board.

Keywords Design-for-security · Side-channel attacks · Fault attacks · Countermeasures · Lightweight · Masking · Shuffling · Fault space transformation · IoT applications side channels · Threshold implenentations

1 Introduction

The advent of the Internet-of-Things (IoT) brings has lent great impetus to the design of *smart devices* that are often embedded with electronics, software, sensors, actuators, and network connectivity for easy communication and interaction. The IoT today is a network of physical objects or things, embedded with electronics, software, sensors, and network connectivity, that can be sensed and controlled remotely across the existing network infrastructure. While IoT creates a number of opportunities for integration of the physical world into computer-based systems for greater economic benefit, it also gives rise to a number of security vulnerabilities that must be handled. In particular, several IoT devices embed cryptographic cores for authentication and information processing, which unless protected, could leak secret information to malicious adversaries. Embedded security solutions for IoT must typically be tuned resource-

The authors would like to thank Intel Corporation, USA for partial funding of the work under the project "LightCrypto: Ultra-Light-weight Robust Crypto-Architectures for Performance and Energy".

✉ Sikhar Patranabis
sikhar.patranabis@iitkgp.ac.in

Debapriya Basu Roy
deb.basu.roy@cse.iitkgp.ac.in

Anirban Chakraborty
anirban.chakraborty@iitkgp.ac.in

Naveen Nagar
naveen009@iitkgp.ac.in

Astikey Singh
astikey@iitkgp.ac.in

Extended author information available on the last page of the article.

constrained environments, and are therefore expected to have low area footprint and low power consumption.

One of the major threats to cryptographic security arises from implementation-based attacks, such as side channels and active fault analysis. A variety of passive side channels, including time, power, and electromagnetic emission [1], have been successfully exploited to recover the secret key from devices running cryptographic algorithms. While most of these attacks are semi-invasive or non-invasive, active invasive attacks such as fault injection [2] can also force a device to leak the secret key as a result of a faulty computation. Both side-channel and fault attacks are easy to mount and do not necessarily require high-end equipment, and are therefore extremely potent threats to cryptographic devices in the present world.

1.1 Background and Related Work

There have been a number of countermeasures proposed in the literature against implementation-based attacks. Countermeasures against side channels can be broadly categorized into three major categories—addition of random noise [3], randomization of state variables via *masking* [4], and derangement in time of sensitive operations via *Shuffling* [5]. While masking provides provable security against power and electromagnetic side-channel attacks, ad-hoc implementation of masking is often too costly and unsuitable for highly resource-constrained devices. Standard shuffling-based countermeasures, on the other hand, are less costly than masking, but are often insufficient as stand-alone countermeasures to resist side-channel attacks. Fault attacks, on the other hand, are often countered via use of redundancy [6], infection [7], or code-based techniques [8]. However, many of these techniques, such as the use of temporal and spatial redundancy, are vulnerable to non-uniform fault distribution-based attacks such as differential fault intensity attacks (DFIA) [9], where the adversary controls the probability of injection of different faults onto the target device [10].

We point out here that many of the aforementioned countermeasure techniques are generic and are often agnostic of the underlying target block cipher, implying that they may not be tailored to suit a particular cipher under test. For instance, while shuffling may be a more optimal choice of countermeasure for lightweight implementations of substitution-permutation network (SPN)-based block ciphers, such as the AES [11] or PRESENT [12], ciphers such as SIMON [13] that do not have substitution boxes more amenable to masking at a much lesser cost. In addition, countermeasures against side channels and fault attacks are often conflicting, in the sense that countering one threat might actually aggravate the other. For instance, the use of robust codes to prevent differential fault analysis (DFA) has been studied in detail in [14], yet it has been found to enhance side-channel vulnerabilities of the design under test owing to the non-linear nature of the code [15]. Finally, fortifying a cryptographic core with a variety of countermeasures without prudent engineering choices imparts a huge overhead on the resources required, especially if not equipped with prudent engineering strategies. This leads to the requirement of lightweight design-for-security methodologies for countering side channels and fault analysis.

1.2 Our Motivation

Our aim in this paper is to propose novel design strategies for cryptographic primitives that make them amenable to lightweight countermeasures for combined resistance against side channels and fault attacks in an IoT environment. The techniques presented in this paper can serve as guidelines for designing cryptographic primitives that are suitable for resource-constrained environments, and are, at the same time, holistically resistant to implementation attacks.

1.3 Countermeasure Strategies and Case Studies in the Paper

In this section, we present an overview of the countermeasure strategies discussed in this paper against combined side-channel and fault analysis, along with the corresponding case studies used to illustrate the effectiveness of these strategies. Table 1 summarizes the overall layout of the paper.

1.3.1 Lightweight Countermeasure Strategy Against Fault Attacks

Fault space transformation (FST), introduced by Patranabis et al. in [16] is a countermeasure strategy against fault attacks that combines traditional redundancy-based techniques with use diffusion layers to hinder the ability of an adversary to inject highly localized/biased faults in multiple computation rounds. In particular, they advocate the use of additional maximum distance separable (MDS) and inverse MDS mappings in the redundant computation to a significant shift in Hamming weight between equivalent faults in the original and redundant computations (for instance, with respect to AES-128, in order to bypass the redundancy check, the adversary must precisely inject a localized single byte fault in the original computation and a distributed four byte fault in the redundant computation, where the four byte fault is the image of the single byte fault under the MDS mapping). However, MDS mappings are typically expensive to implement in lightweight applications. In this paper, we illustrate lightweight hardware-based design strategies for

Table 1 Countermeasure strategies and case studies in the paper

Attack type	Countermeasure strategy	Case-study	Section
Fault attacks	Fault space transformation (MDS matrices)	PRIDE	4
	Fault space transformation (bit permutations)	PRESENT	5
Side-channel attacks	Masking with refresh	PRESENT	8
	Shuffling across rounds		
Fault attacks + side-channel analysis	Fault space transformation + masking + shuffling	PRESENT	9
		GIFT	9

MDS mappings that are suited to area-constrained applications. Note that MDS mappings designed using our strategies serve the dual purpose of diffusion and FST. We validate our techniques using a case study on the recently proposed block cipher PRIDE [17] in Section 4. PRIDE was originally proposed keeping systematic linear layer design for software implementations in mind. However, as demonstrated in the case study, such strategies are not necessarily the most optimal for hardware implementations, especially in FPGA-based applications commonly encountered in IoT. Thus, PRIDE presents us with the ideal framework to illustrate the superiority of our design strategies with respect to hardware implementations. In addition, the case study illustrates the effectiveness of MDS mappings in achieving FST

We then explore a second strategy for FST using bit permutation-based linear layers instead of MDS mappings—an option that was not investigated by Patranabis et al. in [16]. Note that this is immediately attractive from the point of view of lightweight applications, since bit permutations can be implemented efficiently with zero area overhead. In fact, bit permutations are popularly employed in a number of recently proposed lightweight block ciphers, including PRESENT [18] and GIFT [19] However, it provides less optimal diffusion guarantees as compared to MDS layers. For the effectiveness of bit permutations in achieving FST, we make the following probabilistic argument: suppose that the adversary injects a highly localized (single byte/single nibble) fault in the original computation round of a block cipher implementation protected using bit permutation-based FST. Then with very high probability, the equivalent fault to be injected in the redundant computation is distributed across multiple nibbles, and is thus difficult to inject precisely. Additionally, without the exact knowledge of the internal state of the block cipher, the adversary cannot predict the "pathological fault injection instances" where the effect of FST is not sufficient to prevent the fault attack. As a case study, we illustrate the effectiveness of using bit permutations for FST in an implementation of the PRESENT block cipher.

1.3.2 Lightweight Countermeasure Strategies Against Side-Channel Analysis

For resistance against side-channel attacks, we present the following lightweight implementation strategies:

1. **Masking with Periodic Refresh.** Masking of the nonlinear substitution boxes (S-Boxes) [20] is a popular countermeasure against DPA that aims to randomize the S-Box computations, thereby destroying the correlation of the power leakage with the sensitive data during the execution of the block cipher. For ideal security, the mask should be chosen uniformly at random, that is, refreshed after every encryption. This is, however, extremely costly both in terms of area overhead and throughput loss, and is hence rather impractical for lightweight applications targeting resource-constrained devices. As an alternative design strategy, we advocate trading off security with efficiency by refreshing the mask at pre-specified intervals instead of after every encryption. In particular, we focus on serialized block cipher implementations, which is a popular hardware implementation strategy for area-constrained applications.
2. **Choice of S-Boxes.** The periodic refresh strategy is further complemented by a careful choice of S-Box for the block cipher. For this, we resort to using *modified transparency order* (MTO) metric [21] to compare between different S-boxes in terms of their resistance against differential power analysis (DPA). We establish via simulation studies that there exists a direct correlation between the MTO value of a given S-Box, and the adequate refresh rate for a masked implementation of the same to achieve reasonable side-channel security. In other words, a careful S-Box choice guided by the MTO metric potentially reduces the overhead of periodic mask refresh and improves throughput.
3. **Multi-Round Shuffling.** In a shuffled implementation, any key-dependent operation is deranged in time, thereby

increasing the uncertainty of the adversary as to when a specific key-dependent operation occurs in time. This in turn reduces the signal-to-noise ratio (SNR) of the implementation, thereby increasing security against side channels [22]. Standard shuffling practices often advocate shuffling to be restricted within the operations in a single round. This often does not afford sufficient security for shuffling to be used as a stand-alone countermeasure. In this section, we present details of how shuffling may be done across multiple rounds of a substitution-permutation network (SPN) block cipher in a lightweight and efficient manner for improved side-channel resistance.

In order to elucidate the combined effectiveness of the aforementioned strategies in thwarting side-channel attacks, we present a case study involving an implementation of the PRESENT block cipher using a combination of good S-Box choices, fortified via refresh-based masking, and shuffling across two rounds.

1.3.3 Putting Everything Together

We present a final case study on a generalized bit permutation-based PRESENT-like block cipher that combines all our aforementioned design choices for combined resistance against side-channel and fault analysis. More specifically, our implementation combines the following features:

1. Area-efficient FST using bit permutations that provide optimal diffusion across three cipher rounds with high probability
2. MTO-based choice of S-Box (different from the original S-Box of PRESENT) protected using periodically refreshed masks
3. Nibble-wise shuffling of S-Box computations distributed across two cipher rounds

Our implementation is more area-efficient as compared to state-of-the-art implementations of PRESENT that are secure against both side-channel and fault analysis. In term of side-channel leakage, our implementation does not cross the safe leakage threshold against side channel attacks over 50,000 power traces, which appears to be sufficient for a large number of lightweight applications. In addition, it is resistant against both DFA- and DFIA-like fault attacks. We also present an equivalent case study for the block cipher GIFT proposed in [19], where we combine spatial redundancy-based FST using the bit permutation layer of GIFT with a refreshing-based masked implementation of the PRINCE S-Box and shuffling of S-Box operations across rounds.

1.4 The Target Platform: Field Programmable Gate Arrays (FPGAs)

In this paper, we choose FPGAs as the target platform for our case studies owing to their reconfigurability and relevance to IoT. Since IoT applications are often constrained by the dual requirements of high performance and parsimony of resources, reconfigurable hardware seems to be the ideal choice. FPGAs also offer additional advantages, such as low power consumption and low memory bandwidth requirements, making them preferable over GPUs in distributed applications such as image search [23]. Modern FPGAs are equipped with even more sophisticated features, such as dynamic partial reconfiguration (DPR), that allows dynamic, energy-efficient non-invasive modification of the existing circuit on the FPGA, mostly to enhance functionality in the form of added plug-ins. The plug and play philosophy is particularly suitable for IoT applications, since it supports multiple functions at a very large scale without the need for having dedicated hardware available at all times.

1.5 Practical Evaluation of Side-Channel Leakages

In this paper, we use the Test Vector Leakage Assessment (TVLA) methodology [24] to practically evaluate the side-channel leakage of any device under test. TVLA is an extremely popular metric owing to its simplicity, reproducibility and recent international standardization. In particular, we use the non-specific first-order leakage test for computing TVLA on-chip. In this test methodology, two sets of leakage measurements are collected—one with a fixed plain text value and the other with random plaintext values drawn uniformly at random. This is followed by a *statistical t test* that uses the mean and variances of both leakage data sets. The TVLA acts as more of a yes/no metric, in the sense that it certifies a design as either leakage-free or not leakage-free, depending on whether the *t* test statistic exceeds a safe threshold or not.

However, two designs under test may be compared with respect to their side-channel security in terms of the number of leakage samples beyond, which their TVLA value crosses the safe threshold. For instance, if the TVLA value for design-1 crosses the safety threshold for TVLA at 10,000 leakage samples, while that for design-2 crosses the safety threshold beyond 50,000 traces, design-2 can be said to be more resilient to side channel attacks compared to design-1. In addition, if across a certain number of traces, design-2 has a greater TVLA leakage as compared to design-1, the former can be said to be more resilient against side-channel attacks.

2 Lightweight Design Strategies for Countering Fault Attacks on Block Ciphers with MDS Matrices

This section presents our proposed design-for-security strategies to counter fault attacks on block cipher implementations. We begin by briefly reviewing popular fault attack strategies in the literature. We then introduce the concept of FST as a countermeasure against such attacks. Finally, we demonstrate how the linear layer of the block cipher itself may be used efficiently to achieve good FST.

2.1 Fault Attacks on Block Ciphers

The basic principle of any fault attack is to cause a malicious aberration in the normal execution of the target cryptographic algorithm, and to use the corresponding leakage to try and recover the key. In this work, we focus on countering two major varieties of fault attacks—DFA, where the adversary injects a random fault with certain known spatio-temporal characteristics, and analyzes faulty and fault-free ciphertext pairs to recover the secret key, and differential fault intensity analysis (DFIA) that combines principles of side-channel analysis techniques, such as DPA with that of fault based perturbations to recover the secret key from faulty ciphertexts only.

– **Differential Fault Analysis (DFA):** In DFA, the adversary injects a random fault with certain known spatio-temporal characteristics, and analyzes faulty and fault-free ciphertext pairs to recover the secret key.
– **Differential Fault Analysis (DFIA):** DFIA [9] represents a class of fault attacks that combine principles of side-channel analysis techniques, such as DPA with that of fault-based perturbations to recover the secret key from faulty ciphertexts only.

The actual attack procedure and the subsequent key recovery technique depends on the following:

– The nature, or *model* of the fault (such as bit flips, byte faults, stuck-at faults, or random faults) and the nature of information obtained by the adversary (correct and faulty ciphertexts, only faulty ciphertexts or even behavior)
– The spatio-temporal characteristics of the fault (for example, with respect to block ciphers, the precise location and round in which the fault is injected)
– The propagation of the fault to the output of the cryptographic algorithm and the corresponding leakage/ behavior in terms of the information obtained (this usually depends on the various round operations in any block cipher algorithm)

2.1.1 Differential Fault Analysis (DFA)

In DFA, the adversary injects a random fault with certain known spatio-temporal characteristics, and analyzes faulty and fault-free ciphertext pairs to recover the secret key. DFA has been widely studied on a number of block ciphers such as AES [2, 25]. DFA is powerful enough to recover the entire 128 bit key of AES with just a *single* random fault injection [2]. The usage of practically achievable fault models, such as bit faults and byte faults that can be injected using low-cost fault injection techniques [2] makes DFA a potent threat to the security of cryptographic algorithms.

2.1.2 Differential Fault Intensity Analysis (DFIA)

DFIA [9] represents a class of fault attacks that combine principles of side-channel analysis techniques, such as DPA with that of fault-based perturbations to recover the secret key from faulty ciphertexts only. Such attacks require *only* faulty ciphertexts, and exploit the fact that the key-dependent faulty state value has a strongly biased distribution for the correct key hypothesis. This in turn can be distinguished from a random state due to an incorrect key hypothesis using an appropriate distinguisher.

2.1.3 Safe-Error Attacks (SEA), Differential Behavior Analysis (DBA), Fault Sensitivity Analysis (FSA)

Safe error attacks (SEA) and differential behavior analysis (DBA) deduce the presence or absence of a fault during an encryption operation from the *behavior* of a cryptographic device [26, 27]. The crux of the attack lies in the fact that depending on a particular subpart of the secret key (such as a bit or a byte), a fault may or may not lead to a faulty computation. Hence, such key-dependent behavior can be a potential source of leakage, and has in fact been exploited to mount implementation dependent attacks on AES [27].

2.2 Popular Countermeasures Against DFA and Insufficiency against DFIA

Countermeasures against DFA can be broadly classified into the following categories:

2.2.1 Concurrent Error Detection (CED)

CED techniques use four major kinds of redundancies—temporal, spatial, information, and hybrid (combination of space, time, and information) to detect faults. In temporal redundancy, each operation is performed twice, followed by a check to detect errors [28, 29]. A similar concept is that of spatial redundancy, where the system maintains two copies of the same hardware operating in parallel, and checks for errors by comparison [28, 30]. Information redundancy uses additional check bits that are generated by encoding the input message and are transmitted with the encrypted message [6]. The decrypter derives these check bits and checks if they indeed are generated by encoding the decrypted message. These bits could be derived from linear codes, such as parity [31] or non-linear robust codes [14].

2.2.2 Infective Countermeasures

A second variety of countermeasures against fault attacks is based on infection. Infective countermeasures avoid the usage of comparison (as opposed to detection based countermeasures) by diffusing the effect of the fault to render the ciphertext unexploitable. Several propositions regarding infective countermeasures have been made in the cryptographic literature, and the most recently proposed variant [7] combines the usage of redundancy with dummy rounds to confuse the adversary. Infective countermeasures have been proven to be formally secure against standard first-order DFA and have been suitably implemented to prevent even advanced software-based fault attacks such as instruction skips [32]. However, a major shortcoming of infective countermeasures lies in their usage of the additional dummy rounds, which tends to reduce throughput to a large extent.

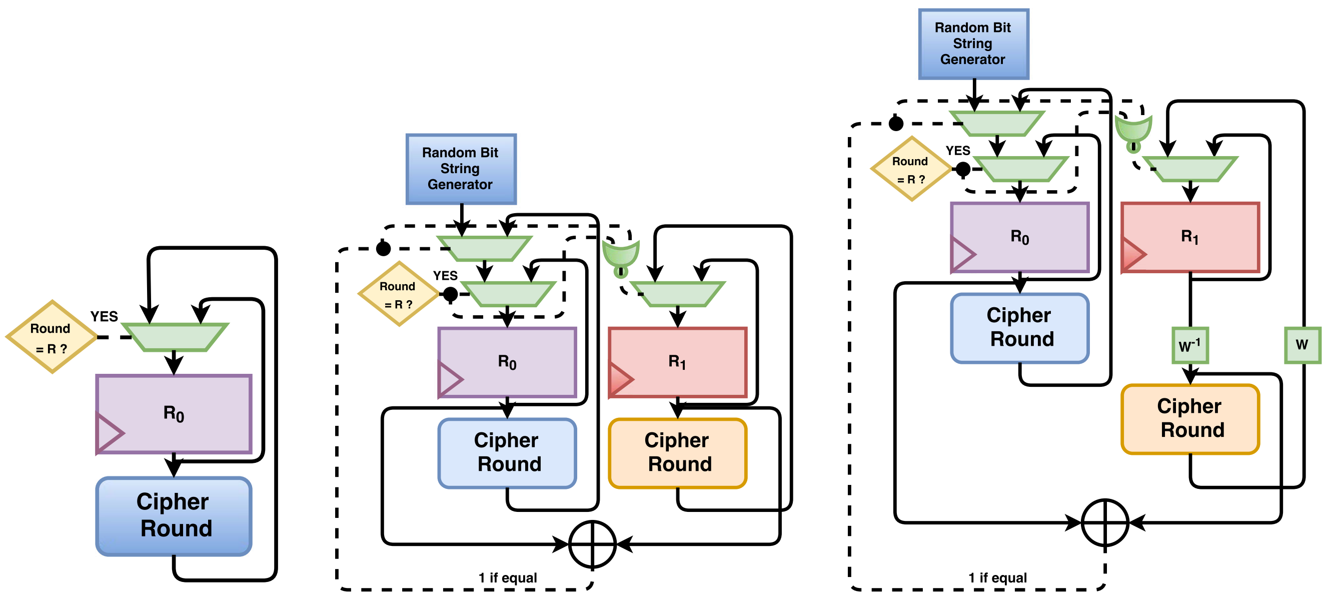
2.2.3 Encoding Based Countermeasures

A third and very efficient class of countermeasures against fault attacks use the concept of *data encoding*. In such countermeasures, the naïve detection step is performed using a different data encoding, so as to reduce the fault collision probability. A foremost example of such technique is the orthogonal direct sum masking (ODSM) [33] that uses linear complementary dual codes to detect and prevent both side-channel and fault attacks. An extended version of this is used to prevent hardware Trojan horse-based fault attacks on encoded circuits [8]. A major disadvantage of the technique in [8, 33] is that it can only detect first-order fault attacks up to a maximum Hamming weight d. Secondly, the usage of masking makes the overhead of ODSM large with respect to fault attack prevention. Another popular encoding technique to prevent fault attacks is the usage of dual rail precharge logic [34].

We note that while each of the aforementioned techniques successfully thwart DFA, they are prone to DFIA due to the biased nature of fault injection in the latter. In fact, as demonstrated by Patranabis et al. in [16, 32], biased fault attacks render most existing redundancy-based countermeasures ineffective, since the adversary can inject the same fault multiple times with reasonably high probability and bypass the detection mechanisms in place. In fact, this holds true even when the fault injection mechanism is not too precise, such as clock and voltage glitches. Additionally, advanced fault attack techniques such as laser injections can also be used to bypass a number of different fault detection mechanisms, including parity check and other code-based techniques [35]. Unfortunately, mechanisms to thwart DFIA had not been studied until a recent work of Patranabis et al. [16], referred to as *fault space transformation*, which acts as a combined countermeasure against DFA and DFIA. We expand more on this technique in the following section. Finally, we would like to point out that there are no known countermeasures against DFIA that appear to be broken by DFA. This may be attributed to the fact that DFIA has been introduced more recently; hence, countermeasures against it are far less studies compared to DFA.

2.3 Fault Space Transformation: a Combined Countermeasure Against DFA and DFIA

Fault space transformation (FST) [16] is a technique that combines the use of redundancy with a transformation between the original and redundant state variables to ensure that the two computations are not identical. While the use of redundancy prevents classical fault attacks such as DFA, the presence of the transformation ensures that biased fault attacks such as DFIA cannot be mounted by introducing identical faults in the original and redundant computations. More specifically, FST provides security against multi-fault injections, where the fault model chosen by the adversary has a localized nature (for example, byte faults in AES-128 or nibble faults in PRESENT-80). The idea behind FST is to ensure (with high probability) a significant shift in Hamming weight between equivalent faults in the original and redundant computations (for instance, with respect to AES-128, in order to bypass the redundancy check, the adversary must precisely inject a localized single byte fault in the original computation and a distributed four byte fault in the redundant computation, where the four byte fault is the image of the single byte fault under the MDS mapping). This shift makes it difficult for the adversary to ensure that an equivalent fault pair is precisely injected in the original and redundant rounds without disrupting the rest of the computation, which is typically difficult to achieve in practice when one of the faults to be injected is highly non-localized. Figure 1 summarizes the concept of FST in conjunction with a time-redundant version of a block cipher. Part (a) demonstrates a simple schematic description for a generalized block cipher structure with no



(a) Unprotected Implementation (b) Spatial Redundancy (c) Spatial Redundancy with FST

Fig. 1 Overview of fault space transformation: combination with spatial redundancy

protection against fault attacks. The register R_0 is updated with the intermediate cipher state value at the end of each round. The block cipher has a total of R rounds. Note that it is assumed that the round function involves the secret key corresponding to a particular round, and an adversary tries to inject a fault in the state register R_0. Part (b) illustrates a spatially redundant version of the same where the hardware for the cipher round is duplicated and the state register is randomized in case the output of the original and redundant operations do not match. Part (c) also incorporates a transformation W between the original and redundant computations, which essentially transforms the fault space and makes equivalent fault injection harder to achieve than in part (b).

2.3.1 Advantages of FST over Code-based Countermeasure Techniques

One of the main reasons we choose FST as a countermeasure technique is because of its two major advantages over code-based countermeasures. First of all, unlike information redundancy-based techniques such as robust codes [14] that use non-linear transformations, FST uses linear transformation, and hence does not affect the side-channel security of the implementation. This is because any linear layer used to implement FST typically comprises entirely of XOR gates, which typically exhibit uniformly balanced switching activity over a large sample of operations. Secondly, unlike techniques such as linear complementary dual codes [33], it provides 100% fault coverage independent of fault properties, such as Hamming weight. Finally, unlike state-of-the-art infective countermeasure techniques [7, 32], it does not require the use of additional dummy rounds, and hence affords better throughput.

2.4 FST and the Linear Layer: Choosing the Transformation *W*

The authors of [16] propose the use of *maximum distance separable* (MDS) matrices [36] for W. An MDS matrix is a matrix representing a function with special diffusion properties and has many useful applications in cryptography, especially in designing multi-permutations to prevent cryptanalysis. Now, suppose that the linear transformation W is a $m_2 \times m_1$ MDS mapping over a field K from K^{m_1} to K^{m_2}. MDS FST redundant fault spaces $\mathcal{F}_0$ and $\mathcal{F}_1$ differ sufficiently in their Hamming weights to have low correlation of occurrence. Let the adversary inject a t byte fault f_0 in the register R_0, and let f_1 be the corresponding fault to be injected in the register R_1 so that the countermeasure fails to detect the fault injection. By the MDS diffusion property, any t byte fault f_0 is mapped an $m_2 - t + 1$ byte fault f_1. For the special case of a single byte fault ($t = 1$), the transformed fault space comprises of faults that affect at least m_2 bytes of the output. Thus, the precision of the transformed fault space $\mathcal{F}_1$ is approximately $\frac{1}{2^{(8m_2-1)}}$ times lower than the original fault space $\mathcal{F}_0$, making it difficult for the adversary to create equivalent fault injections with high probability. Additionally, since using MDS matrices causes W to be a linear transformation, the side-channel leakage of the implementation is not adversely affected [15].

It seems that incorporating FST in a block cipher design would lead to huge hardware overhead due to the requirement of a suitable MDS transformation. We point out here, however, that the use of MDS matrices for designing linear layers of block ciphers is a widely accepted strategy in today's literature, since it guarantees a suitably active number of S-boxes across a certain number of rounds, and hence good linear and differential characteristics. The same MDS transformation present in the linear layer of the block cipher can hence serve the dual purpose of resisting cryptanalytic attacks and fault attacks, thereby eliminating the overhead due to an extra linear transformation for FST.

The resource-constrained environments frequently encountered in IoT-based applications demand that any design strategy for crypto-primitives, such as block ciphers must ensure a low area footprint. Given that even the most optimized construction of non-linear S-Boxes [37, 38] imposes huge penalties on the gate count of hardware implementations, the construction of lightweight linear layers with good diffusion properties is of paramount importance to the design of low energy ciphers. However, to the best of our knowledge, the systematic study of design strategies for such optimal linear layers is scarce in literature. Most current designs for linear layers are either ad-hoc [39], or focus mostly on strong security arguments against linear and differential cryptanalysis, but are agnostic on the hardware requirements of the designs.

3 Lightweight MDS Layer Design: Combining Linear Layer Design with Fault Space Transformation

In this section, we focus on lightweight design strategies for linear layers with MDS properties that ensure resistance against both cryptanalytic attacks and fault attacks. We first point out that the space of MDS matrices is too huge to make the exhaustive search of suitable lightweight transformations practically infeasible. Designers of lightweight crypto-primitives such as PHOTON [40] and LED [41] have specifically chosen MDS matrices with lightweight roots, which could be iterated multiple times to arrive at the required MDS mapping. However, there is no discussion on how such a strategy can be generalized into a design guideline that narrows down the search space within practical limits. Existing research on identifying MDS transformations with lightweight roots is not guaranteed to be optimal and is computationally intensive [42]. A tangible solution has been presented in the construction of the block cipher PRIDE [17], which uses block interleaving, and has been claimed by the authors to be efficient in terms of both software and hardware implementations, although no hardware design for the same has been reported to the best of our knowledge. Finally, [43] provides new methods to look for lightweight MDS matrices, and in particular involutory ones. By proving many new properties and equivalence classes for various MDS matrices constructions, such as circulant, Hadamard, Cauchy, and Hadamard-Cauchy, the authors of [43] exhibit new search algorithms that greatly reduce the search space and make lightweight MDS matrices of rather high dimension possible to find. Follow-ups to this work have focused on lightweight circulant involutory MDS matrices [44], lightweight generalized circulant matrices [45], [46], optimized implementations [47], and almost MDS matrices over rings with zero divisors [48].

3.1 Notations Used

In this section, we introduce the basic notations used in this paper. We denote by F_2 the field with two elements and by F_2^b the extension field of F_2, obtained using an irreducible polynomial of degree b. We also denote by $(F_2)^n$ the n-dimensional vector space over F_2, and by $(F_2^b)^n$ the n-dimensional vector space over the corresponding extension field F_2^b.

In literature, the weight of a vector $x = (x_1, x_2, \cdots, x_n) \in (F_2^b)^n$ (where each $x_i \in F_2^b$) is given by $\mathrm{wt}_b(x) = |\{1 \leq i \leq n | x_i \neq 0\}|$. Consequently, given a linear mapping $L : (F_2^b)^n \longrightarrow (F_2^b)^n$, its differential branch number $\mathcal{B}_d(L)$ and linear branch number $\mathcal{B}_l(L)$ are given by $\mathcal{B}_d(L) = \min\{\mathrm{wt}_b(x) + \mathrm{wt}_b(L(x)) | x \in (F_2^b)^n, x \neq 0\}$ and $\mathcal{B}_l(L) = \min\{\mathrm{wt}_b(x) + \mathrm{wt}_b(L^*(x)) | x \in (F_2^b)^n, x \neq 0\}$ respectively. Here, L^* denotes the adjoint linear mapping of L, which, in this context refers to the transposed matrix of L. The differential and linear branch numbers of the linear mapping are cryptographically significant because these can be used to define upper bounds on the differential and linear characteristics of the cipher.

In particular, greater the differential and linear branch numbers, the lower is the average probability that the adversary can obtain a differential or linear trail. Thus, in order to achieve the desirable *diffusion* properties, the linear mapping L must satisfy a minimum differential and linear branch number. From the perspective of coding theory, the differential branch number corresponds to the minimal distance of the F_2-linear code C over F_2^b with generator matrix $G = [I|L^T]$, where I is the $n \times n$ identity matrix. In other words, C is a $(2n, 2^n)$ additive code over F_2^b with minimal distance $d = \mathcal{B}_d(L)$. Analogously, the linear branch number corresponds to the minimal distance of the

F_2-linear code $C^{\perp}$ over F_2^b with generator matrix $G^* = [L|I]$. Note that C and $C^{\perp}$ are dual codes and do not necessarily have the same minimum distance.

Suppose that a matrix $G = [I|L]^T$ is the generator matrix for a F_2-linear $(2n, 2^n)$ code with minimal distance d. Then, L has *at least* $d-1$ *ones per row*. The proof for this statement is fairly straightforward. A similar proof can also be given for the statement that if $G^* = [L|I]$ is the generator matrix for a F_2-linear $(2n, 2^n)$ code with minimal distance d, then L has *at least* $d-1$ *ones per column*.

We now present a construction strategy for the linear layer of an SPN block cipher, with focus on minimizing the hardware footprint of the cipher, while guaranteeing desirable cryptographic security.

3.2 Combining Lightweightedness with MDS Properties

Designing MDS matrices that are lightweight in terms of hardware footprint is a major challenge. The high diffusion property provided by MDS matrices is due to their high linear and differential branch numbers, which in turn puts a lower bound on the number of ones in each row and each column. In particular, any $m \times m$ binary MDS matrix with linear and/or differential branch number d, must have at least $m(d-1)$ ones in the entire matrix. This leads to a trade-off between lightweightedness and diffusion properties. A solution to this is to choose MDS matrices with lightweight roots that have less hardware requirements, and can then be iterated to obtain the necessary diffusion properties. However, for a given dimension $m \times m$ and branch number d, there are as many as $(\binom{m}{d-1}2^{m-d+1})^m$ such matrices. For instance, let us consider $m = 32$ and $d = 5$, which is the case for the linear layer of PHOTON. In this case, the space is approximately $2^{28\times 32}$, which is enormously large for a brute force search. An alternative strategy to this is to start from the underlying lightweight matrices and arrive at the desired MDS matrices by iterating them. Further, we reduce the search space for these underlying matrices by assuming a simplified form for them, as described below.

We assume that the matrices are defined over some GF(2^t) field. The basic technique is to start with an underlying matrix A of either of the two forms as shown below.

$$\begin{pmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \\ Z_1 & Z_2 & \cdots & Z_m \end{pmatrix}, \begin{pmatrix} Z_1 & \cdots & Z_{m-1} & Z_m \\ 1 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{pmatrix}$$

and then iterate to arrive at the desired MDS matrix (here 0, 1 and $Z_i \in GF(2^t)$ for $i = 1, 2, \cdots, n$). Since the actual hardware implementation contains only A, we achieve massive savings in terms of gate count. The choice of elements in the last row of A is governed by two major factors—the compactness of A, and the linear and differential characteristic of the final MDS matrix A^j. The main challenge in this design strategy is to tackle the search space for the final row, which has size $(2^t)^m$ and thus makes an exhaustive search difficult for large values of m.

We point out here that although we refer to matrices over GF(2^t), each such matrix also has an equivalent binary representation, denoted as the *companion matrix* representation [49]. In the companion matrix, each GF(2^t) element in the original matrix by a corresponding $t \times t$ binary sub-matrix. Thus, any $m \times m$ matrix comprising of GF(2^t) elements has a corresponding mt $\times$ mt companion matrix representation. *In this work, we define the lightweightedness of a GF(2^t) matrix in terms of the density of its companion matrix.*

For the sake of completeness, we briefly review the fundamentals of companion matrix representation next. We then move on to our proposed construction methodology.

3.3 The Companion Matrix Representation

A large number of ciphers such as AES, TwoFish and PHOTON use GF(2^t) multipliers for MDS matrix multiplications. In a GF(2^t) multiplier, each GF(2^t) element has a $t \times t$ companion matrix representation over F_2, depending on the primitive polynomial $p(x)$ of degree t used to construct the field. We present a small example here to elucidate the basic construction idea. Let $a = 3$ be an element in GF(2^8), where the primitive polynomial is given by $p(x) = x^8 + x^4 + x^3 + x + 1$. Now let $\mathcal{T}_{a,p}$ be the corresponding 8×8 companion matrix representation of a. The basic construction steps are as follows:

1. The first column vector or col[0] is given by the 8-bit binary representation of a, which is $(00000011)_2$
2. Every other column vector i for $i = 1, \cdots, 7$ is obtained by left shifting column $i-1$, and XOR-ing the resultant vector with the polynomial $p(x)$ in the event of an overflow.

The final companion matrix $\mathcal{T}_{a,p}$ is given by the following:

$$\mathcal{T}_{a,p} = \begin{pmatrix} 1&0&0&0&0&0&0&1 \\ 1&1&0&0&0&0&0&1 \\ 0&1&1&0&0&0&0&0 \\ 0&0&1&1&0&0&0&1 \\ 0&0&0&1&1&0&0&1 \\ 0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&1&1&0 \\ 0&0&0&0&0&0&1&1 \end{pmatrix}$$

Our proposed linear layer design strategy uses this concept to design $\frac{n}{t} \times \frac{n}{t}$ matrices over $GF(2^t)$, such that the corresponding $n \times n$ companion matrices are MDS in nature, that is, they satisfy the necessary linear and differential properties to be secure enough against linear and differential cryptanalysis.

3.4 Construction of the Linear Layer : Block Interleaving

We now concentrate on our proposed strategy for constructing the linear layer using the wide-trail technique. We assume that we are dealing with SPN block ciphers, such that the substitution layer consists of n S-box instances, each of dimension b acting in parallel. We intend to construct the linear layer $\mathcal{L}$ for this cipher. In order to achieve the desired diffusion properties, $\mathcal{L}$ must be constructed such that the matrix $\mathcal{G} = [I|\mathcal{L}^T]$ generates a $(2n, 2^n)$ additive code with minimal distance d over F_2^b. The linear layer $\mathcal{L}$ is broken up into two major components—the *permutation mapping* and the *MDS mapping*. We denote by $P^n_{b_1,\cdots,b_k}$ the permutation matrix that maps $(x_1, \cdots, x_n) \in ({F_2}^{b_1} \times {F_2}^{b_2} \times \cdots \times {F_2}^{b_k})^n$ to $((x_1^{(1)}, \cdots, x_n^{(1)}), \cdots, (x_1^{(k)}, \cdots, x_n^{(k)})) \in ({F_2}^{b_1})^n \times ({F_2}^{b_2})^n \times \cdots \times ({F_2}^{b_k})^n$, where $\sum_{i=1}^k b_i = b$. The MDS mapping, denoted by L, is constructed as a direct sum of k smaller MDS mappings $L_1, L_2, \cdots, L_k$, such that L_i has dimensions $\mathrm{nb}_i \times \mathrm{nb}_i$. Quite evidently, L has dimension $\mathrm{nb} \times \mathrm{nb}$, and may be represented as follows :

$$L = \begin{pmatrix} L_1 & \varnothing & \varnothing & \varnothing & \cdots & \varnothing \\ \varnothing & L_2 & \varnothing & \varnothing & \cdots & \varnothing \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \varnothing & \varnothing & \varnothing & \varnothing & \cdots & L_k \end{pmatrix}$$

where $\varnothing$ is all zero binary matrices of appropriate dimensions. This kind of a construction is popularly referred to in literature as *block interleaving* [50]. Now, for each i, let ${G_i}^n = [I|{L_i}^T]$ be the generator matrix for a F_2-linear $(2n, 2^n)$ code with minimal distance d_i over ${F_2}^{b_i}$ for $1 \leq i \leq k$. Then, the matrix $\mathcal{G} = [I|\mathcal{L}^T]$ with $\mathcal{L} = (P^n_{b_1,b_2,\cdots,b_k})^{-1} \circ L \circ P^n_{b_1,b_2,\cdots,b_k}$ is the generator matrix of a F_2-linear $(2n, 2^n)$ code with minimal distance d over ${F_2}^b$, where $d = \min_i d_i$. For a detailed proof of this statement, refer [17]. A similar argument can also be formed for the dual code $C^\perp = [\mathcal{L}|I]$ to ensure that $\mathcal{L}$ also has a necessary minimal distance with respect to the linear characteristic. Further, it is easy to see that the linear and differential characteristics of the overall linear layer $\mathcal{L}$ is the same as that of the MDS mapping L, while the permutation mapping (along with its inverse) essentially enhance the diffusion properties. We now focus on how to choose the smaller MDS mappings L_i such that they achieve the necessary diffusion and can also be implemented in a lightweight fashion. Figure 2 summarizes the construction methodology discussed here.

3.5 A Bottom Up Strategy: Construction of MDS Layers Using Smaller Matrices

We now concentrate on our proposed strategy for constructing the MDS linear layer for a block cipher using the wide-trail technique. Our approach is to construct the MDS mapping, denoted by L, as a direct sum of k smaller MDS mappings $L_1, L_2, \cdots, L_k$, such that L_i has dimensions $\mathrm{nb}_i \times \mathrm{nb}_i$. Quite evidently, L has dimension $\mathrm{nb} \times \mathrm{nb}$, and may be represented as follows :

$$L = \begin{pmatrix} L_1 & \varnothing & \varnothing & \varnothing & \cdots & \varnothing \\ \varnothing & L_2 & \varnothing & \varnothing & \cdots & \varnothing \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \varnothing & \varnothing & \varnothing & \varnothing & \cdots & L_k \end{pmatrix}$$

where $\varnothing$ is all zero binary matrices of appropriate dimensions. This kind of a construction is popularly referred to in literature as *block interleaving* [17]. We next focus on how to design each of the sub-matrices $L_1, L_2 \cdots L_k$. The general idea is as follows. Let d be the minimum branch number that we wish the MDS matrix L to have. Then, each sub-matrix L_i must have a branch number greater than or equal to d, which is equivalent to having at least d ones in each row and each column. Now, each binary sub-matrix L_i can be viewed as the companion matrix representation of the corresponding MDS matrix $L_{i,2^t}$ over $\mathrm{GF}(2^t)$ with

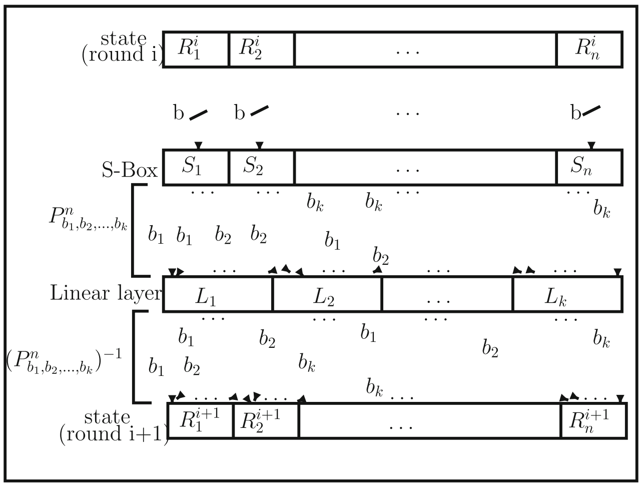


Fig. 2 The construction technique

dimension $\frac{n}{t} \times \frac{n}{t}$. Each matrix $L_{i,2^t}$ is constructed using the iterative technique described in Section 3.2, that is $L_{i,2^t} = (A_{i,2^t})^c$ for some constant c and a lightweight matrix $A_{i,2^t}$ over GF(2^t). Note that the exhaustive search space for the last row of each $A_{i,2^t}$ is $(2^t)^{\frac{n}{t}} = 2^n$, as opposed to $(2^t)^{\frac{nk}{t}} = 2^{nk}$ in case the entire matrix L were to be constructed in this fashion. Thus, the divide and conquer approach makes it much more computationally efficient to choose the underlying lightweight matrices. The appropriate choice of the underlying matrices is now made by a brute force search in the smaller space, with focus on minimizing the number of iterations c as well the hardware footprint of the lightweight companion matrices A_i.

3.6 Ensuring High Dependency

An additional requirement that a well-designed linear layer needs to satisfy is high *dependency* [17]. An S-box i is said to be dependent on an S-box j if an input differential to S-box i in a given round leads to an input differential for S-box j in the next round. Since ensuring a high branch number does not ensure a high dependency, we need to address the two issues independently. For instance, in our proposed construction, we could have $L_1 = L_2 = \cdots = L_k$ such that they have the minimum possible hardware overhead as well the desired branch number d. However, this would lead to poor dependency because each position (j_1, j_2) such that $L_i(j_1, j_2) = 0$ will cause the S-boxes j_1 and j_2 to be totally independent. We propose a strategy to address this issue. In Section 3.2 we showed two different forms for the underlying lightweight matrices $A_{i,2^t}$. We further point out here that each lightweight matrix $A_{i_1,2^t}$ of the first form has a corresponding lightweight matrix $A_{i_2,2^t}$ of the second form, such the last row of the former serves as the first row of the latter. These matrices thus have identical hardware footprint and, when iterated, result in corresponding MDS matrices $L_{i_1,2^t}$ and $L_{i_2}, 2^t$ with identical branch numbers. Further, we have $L_{i_1,2^t}(j_1, j_2) = L_{i_2,2^t}(\frac{n}{t} - j_1, \frac{n}{t} - j_2)$. The probability that both these entries are 0 is now $\frac{1}{4}$. Thus, alternating between the two forms helps to improve dependency without adversely affecting either the branch number or the lightweightedness.

It is easy to see that the inverse of the MDS matrix constructed above has the same differential and linear characteristics as the original matrix, and can be implemented in a lightweight manner. The first property is true for any matrix, since it follows from simple coding theory arguments. The second property, while not true for any arbitrary matrix, holds for our construction methodology. In particular, the inverse MDS matrix is also guaranteed to have a root that can be analogously constructed from smaller lightweight matrices.

3.7 The Inverse Linear Layer

When designing the linear layer for a block cipher, it is important to keep in mind that the inverse linear layer should also be lightweight, and have the same diffusion characteristics. Of these, the second property is guaranteed because, if a matrix L is an MDS matrix with minimum distance d, then its inverse is also an MDS matrix with minimum distance d. The first property, however, is more difficult to achieve. For instance, the inverse MixColumns operation of AES is much more difficult to implement in a lightweight fashion than the MixColumns operation [51]. However, our proposed lightweight construction ensures that both of the abovementioned properties are satisfied for the inverse matrix as well. To show this, it is sufficient to show that the inverse of each sub-matrix, given by $(L_{i,2^t})^{-1}$, and its corresponding lightweight root $(A_{i,2^t})^{-1}$ satisfy these properties. Let $A_{i,2^t}$ be a lightweight matrix of the first form described in Section 3.2. Then its inverse $(A_{i,2^t})^{-1}$ is as follows:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ Z_1 & Z_2 & Z_3 & Z_4 & \cdots & Z_n \end{pmatrix}, \begin{pmatrix} Y_1 & Y_2 & Y_3 & Y_4 & \cdots & Y_n \\ 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

where $Y_n = (Z_1)^{-1}$ and $Y_i = (Z_1)^{-1} Z_{i+1}$ for $i = 1, 2, \cdots, (n-1)$. This means $(A_{i,2^t})^{-1}$ is essentially a lightweight matrix of the second form. Analogously, a lightweight matrix of the second form has a lightweight inverse of the first form. Thus, the inverse linear layer is also always implementable in a lightweight manner. Note that this is a vital observation with respect to cryptographic primitives such as block ciphers.

3.8 An Illustration of the Proposed Strategy

To illustrate the efficiency of the proposed construction technique, we present an instance of a 4×4 MDS matrix L over GF(2^8) with target differential and linear branch number 5 constructed using our technique. The recursive construction first chooses a pair of 2×2 lightweight matrices $A_{1,2^8}$ and $A_{2,2^8}$ such that the number of iterations required is minimum (4 in this case). The corresponding search space is $(2^8)^2 = 2^{16}$, and hence can be explored. The descriptions of the smaller matrices are as follows:

$$A_{1,2^8} = \begin{pmatrix} 0 & 1 \\ 1 & 3 \end{pmatrix} \quad A_{2,2^8} = \begin{pmatrix} 3 & 1 \\ 1 & 0 \end{pmatrix}$$

Iterating the matrices 4 times, the following 2×2 MDS matrices are obtained:

$$L_{1,2^8} = \begin{pmatrix} 4 & 15 \\ 15 & 21 \end{pmatrix} \quad L_{2,2^8} = \begin{pmatrix} 21 & 15 \\ 15 & 4 \end{pmatrix}$$

Using our construction, as proposed in Section 3.5, the final 4×4 MDS matrix L_{2^8} is generated:

$$L_{2^8} = \begin{pmatrix} 4 & 15 & 0 & 0 \\ 15 & 21 & 0 & 0 \\ 0 & 0 & 21 & 15 \\ 0 & 0 & 15 & 4 \end{pmatrix}$$

As discussed, obtaining the companion binary matrices for each element of the field GF(2^8), expands the matrix L_{2^8} into a binary matrix L of dimension 32×32. Now, we compare this matrix with that of PHOTON [40], which we refer to as L'. Matrices L and L' have differential/linear branch numbers of 5 and 4 respectively, and the underlying lightweight matrices have 70 and 62 ones respectively. Furthermore, an attempt to build a 4×4 MDS matrix directly from underlying lightweight matrices of the form described in Section 3.2 leads to a search space of 2^{32}, which is significantly larger than the search space of 2^{16} in our approach. Thus, our proposed methodology yields cipher linear layers with similar area footprint, and good diffusion properties (i.e., both branch number and dependency), while also reducing the computational complexity of the construction process.

4 Case Study 1: Combining Linear Layer Design with FST on a PRIDE-like Block Cipher

In this section, we present a case study based on the block cipher PRIDE proposed recently in [17]. The linear layer of PRIDE also uses block interleaving, but the sub-matrices are chosen to be circulant matrices. In our case study, we substitute the original linear layer of PRIDE with a linear layer constructed using our proposed technique. The interleaving construction is used, with each sub-matrix populated using GF(2^8) elements that are chosen according to the strategy proposed in Section 3. The diffusion matrix L for our modified PRIDE is constructed by interleaving four binary MDS matrices, each of dimension 16×16. Each of these matrices has differential and linear branch number equal to 4, which implies that the overall matrix also has the same linear and differential branch numbers, as in original PRIDE. These matrices are obtained by exhaustively searching the space of 2×2 lightweight matrices of the form discussed in Section 3.2, with focus on minimizing hardware requirements and maximizing dependency properties. We first briefly describe the linear layer to be used, and then compare the hardware costs of our construction with that of the original construction. We point out that, although the authors of PRIDE claim that PRIDE is hardware efficient, no results related to hardware implementations of PRIDE are available in literature to the best of our knowledge.

4.1 The Modified Linear Layer

4.1.1 The Permutation Mapping

The permutation layer P chosen for our proposed cipher is the bit wise permutation matrix $P^{16}_{1,1,1,1}$, as in the original construction for PRIDE. This permutation layer is suitable for hardware implementations as it comes entirely free of cost.

4.2 The Modified MDS Layer

The diffusion matrix L is constructed by interleaving four binary MDS matrices, each of dimension 16×16. Each of these matrices has differential and linear branch number equal to 4, which implies that the overall matrix also has the same linear and differential branch numbers, as in original PRIDE. These matrices are obtained by exhaustively searching the space of 2×2 lightweight matrices of the form discussed in Section 3.2, with focus on minimizing hardware requirements and maximizing dependency properties.

The diffusion matrix L is constructed using the interleaving construction. We construct 4 binary MDS matrices L_1, L_2, L_3, and L_4, each of dimension 16×16 and combine them to obtain the final 64×64 matrix L.

$$\begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}^4 = \begin{pmatrix} 5 & 8 \\ 8 & 21 \end{pmatrix}, \begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix}^4 = \begin{pmatrix} 21 & 8 \\ 8 & 5 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 4 \end{pmatrix}^4 = \begin{pmatrix} 17 & 64 \\ 64 & 10 \end{pmatrix}, \begin{pmatrix} 4 & 1 \\ 1 & 0 \end{pmatrix}^4 = \begin{pmatrix} 10 & 64 \\ 64 & 17 \end{pmatrix}$$

4.2.1 Security Against Classical Cryptanalysis

Since the differential and linear branch numbers of each of the four sub-matrices L_0, L_1, L_2, and L_3 is 4, it is 4 for the entire linear layer L by construction. From the MDS property, this implies that a minimum of 4 S-boxes are active every 2 rounds, and 32 in 16 rounds. As in PRIDE, the choice of S-boxes implies that there is no single differential trail over 16 rounds with average probability better than $(\frac{1}{4})^{32} = 2^{-64}$. Similarly, for linear cryptanalysis. the maximum absolute correlation value for a single linear trail is upper bounded by $(\frac{1}{2})^{32} = 2^{-32}$. Since our proposed cipher has the same differential and linear characteristics as PRIDE, the security analysis for PRIDE against round reduced implementations, as well as other forms of attacks such as algebraic attacks also holds for our cipher.

4.2.2 Hardware Implementation Results

We implemented both the original and modified versions of on a Virtex-5 FPGA (XC5VLX330T) and compare their resource consumptions. To the best of our knowledge, this is the *first report in literature on an FPGA implementation of PRIDE*. The architecture of PRIDE with our proposed linear layer construction is shown in Fig. 3. This architecture is an example of iterative style of implementation where a single round implementation is iterated multiple times to execute the cipher operation. The architecture of the original PRIDE is nearly same as Fig. 3, with the only difference being in the MDS layer implementation. Table 2 presents a detailed summary of the design of the original version of PRIDE on FPGA, while Table 3 presents the corresponding details for modified PRIDE. Table 4 then compares the resource requirements for the linear layer of original and modified PRIDE in terms of gate counts and LUTs. The gate equivalent values are presented in terms of 2-input XOR gates. We note that the modified design does not increase the resource requirements for any other component except the control unit, which requires 6 LUTs more than

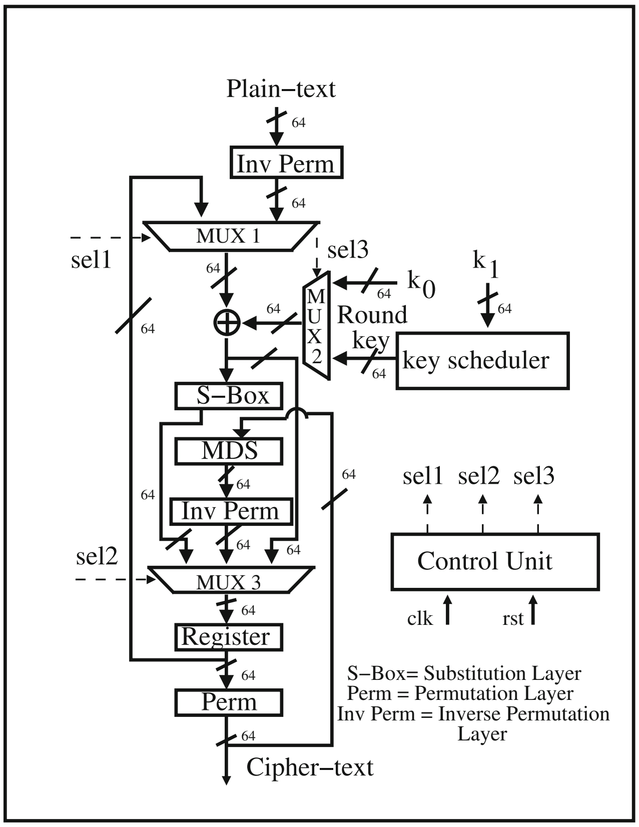


Fig. 3 The architecture of PRIDE with proposed design specification

the original PRIDE. The additional control unit overhead in modified PRIDE stems from the need for additional multiplexers when iterating over the lightweight roots of the MDS matrices to achieve the same linear and differential characteristics as original PRIDE. The overall savings for the full implementation in modified PRIDE, as compared to original PRIDE, is thus 10% (26 LUTs), which is 40% of the *maximum savings* that could be achieved in any hardware implementation in PRIDE without altering any of the other components.

4.2.3 Description of the Architecture for Original PRIDE

The architecture of PRIDE with our proposed linear layer construction is shown in Fig. 3. This architecture is an example of iterative style of implementation where a single round implementation is iterated multiple times to execute the cipher operation. The architecture of the original PRIDE is nearly same as Fig. 3, with the only difference being in the MDS layer implementation. A detailed summary of the design of the original version of PRIDE on FPGA is presented in Table 2. It is important to note that the LUT requirement for linear layer of PRIDE is approximately 25% of the total LUT requirements. Thus, keeping the rest of the architecture including the S-Box and the key schedule intact, the *maximum total LUT savings* that any optimization of the linear layer might yield is *upper bounded* by 25%. Nonetheless, any saving is crucial from the point of lightweight cryptography.

4.2.4 The Resource Savings Achieved in Hardware

We next describe the resource savings achieved by our design methodology for the linear layer of PRIDE. We present our results in terms of both gate savings in ASIC, as well as LUT savings on FPGA. It is important to note that the expected savings with respect to gate counts on ASIC does not translate exactly to LUT savings on FPGA. This difference however can be explained mathematically using the formulation presented in [52] as follows. Consider a q bit combinatorial circuit. Then, the minimum number of LUTs required to implement this circuit can be lower bounded as follows:

$$\#\mathrm{LUT}(q) = \begin{cases} 0 & \text{if } q = 1 \\ 1 & \text{if } 1 < q \leq 4 \\ \lceil x \rceil & \text{if } q > 4 \text{ and } q \bmod 3 = 2 \\ \lfloor x \rfloor & \text{if } q > 4 \text{ and } q \bmod 3 \neq 2 \end{cases} \quad (1)$$

Table 4 compares the resource requirements for the linear layer of PRIDE in terms of gate counts and LUTs. The gate equivalent values are presented in terms of 2-input XOR gates. We note that the modified design does not increase the resource requirements for any other component except the

Table 2 Resource requirement of original PRIDE on Virtex-5 FPGA

Design name	Design module	LUT	Flip-flops	DSP blocks	Clock cycles	Maximum frequency
	Key-Xor+MUX 1 MUX 2	64	0	0		
	S-Box	64	0	0		
	MDS	64	0	0	22	163 MHz
Original pride	MUX 3	64	0	0		
	Control unit	7	5	0		
	Key schedule	0	0	4		
	Register	0	64	0		
	Total	263	69	4		

control unit, which requires 6 LUTs more than the original PRIDE. The overall savings for the full implementation is thus 10%, which is 40% of the *maximum savings* that could be achieved in any hardware implementation in PRIDE without altering any of the other components.

Since each round requires a single clock cycle, the overall implementation of PRIDE has a requirement of 22 clock cycles, including two additional clock cycles for the key whitening operations. Our modified version of PRIDE, on the other hand, must iterate the MDS part of the linear layer four times in each round except the penultimate round. However, we point out that the critical path delay of the linear layer of PRIDE is four 2-input XOR gates due to the presence of a minimum of five ones in each row of the linear layer matrix. On the other hand, the modified linear layer designed using our technique uses extremely lightweight underlying matrices with a *maximum of three ones* in any row. The critical path delay is thus due to two XOR gates. On FPGA, according the formulation presented in [52], the maximum combinatorial delay of a q-bit circuit is given by $\lceil \log_4(q) \rceil$ times the delay of a single LUT. This value is 4 and 2.5 respectively for the original and modified linear layer designs. Hence, for the modified PRIDE, we perform the linear layer operation using a faster clock with double the frequency of the original clock used for the other operations, such as substitution and key-XOR. This clever use of clock switching between the non-linear and linear layer operations leads to an overall requirement of 51 clock cycles for the modified PRIDE. Now, if we consider the area savings in terms of gate requirement (60% on ASIC), the overall *area time product* for the modified PRIDE is *less by* 7.5%.

4.3 Resistance to Fault Attacks

In this section, we examine the fault tolerance of an implementation of PRIDE that uses its linear layer for FST space transformation in a time-redundant fashion. Our approach is to inject highly localized faults within a single nibble, and to gradually enhance the fault intensity so as to increase the number of faulty bits within the target nibble. We illustrate the difficulty of the attack in the presence of FST by presenting the distribution of faults in the original round and the corresponding equivalent fault in the redundant round in Fig. 4. The fault injection parameters are the same for both the injections. We note that in the presence of FST, all localized single nibble faults in the original computation are mapped to faults spreading across four nibbles in the redundant computation, all two nibble faults are mapped to either four or eight nibble faults, all

Table 3 Resource requirement of modified PRIDE on Virtex-5 FPGA

Design name	Design module	LUT	Flip-flops	DSP blocks	Clock cycles	Maximum frequency
	Key-Xor+MUX 1 MUX 2	64	0	0		
	S-Box	64	0	0		
	MDS	32	0	0	45	185 MHz
Modified pride	MUX 3	64	0	0		
	Control unit	13	10	0		
	Key schedule	0	0	4		
	Register	0	64	0		
	Total	237	74	4		

Table 4 Comparison of original and modified PRIDE : hardware resources

Design platform	Original PRIDE	Modified PRIDE	Percent savings
ASIC	128 2-input XORs	50 2-input XORs	60
FPGA	64 LUTs	32 LUTs	50

three nibble faults are mapped to either four, eight, or twelve nibble faults. Beyond this, the equivalent fault spreads across all the nibbles of the cipher state in the redundant computation. Using low-cost fault injection techniques such as clock glitches, and even more sophisticated techniques such as EM pulse injections, such an attack is practically very difficult to achieve.

5 Are MDS Transformations Mandatory for Achieving FST?

In the aforementioned discussion, we have illustrated a design strategy for obtaining MDS transformations with lightweight roots that serve the dual purpose of diffusion (for resistance against classical cryptanalysis) and FST (for resistance against fault attacks). However, there exist a number of block ciphers that do not use MDS transformations, examples being—SIMON, SPECK, and PRESENT. Some of these block ciphers, such as PRESENT, have specially designed linear layers so as to ensure the adequate linear and differential branch numbers for a desired number of active S-Boxes across a certain number of rounds. In others, such as SIMON, the linear layer merely consists of bit shifts and XOR gates. It appears that incorporating FST in such block cipher designs would require the introduction of an additional MDS transformation, thereby compromising to a large extent the lightweight nature of the design itself. However, this is not necessarily true. In the following case study on PRESENT, we illustrate how its bit permutation - based linear layer suffices to achieve FST, thus doing away with the need for an additional MDS block in the design.

5.1 Case Study 2: FST Using Non-MDS Linear Layer of PRESENT

We present a second case study using PRESENT to illustrate that even non-MDS FST injection methodology is identical to that described in Section 4. The target is the block cipher PRESENT, augmented with spatial redundancy-based FST using its bit permutation-based linear layer. We illustrate this by presenting the distribution of faults in the original round and the corresponding equivalent fault in the redundant round in Fig. 5. Quite surprisingly, the distribution of faults, presented in Fig. 5, are nearly identical to that in Fig. 4 for PRIDE. The observation may be explained as follows. The bit permutation layer of PRESENT ensures that nibble of the input of the permutation branches out into four different nibbles of the output—a property that most MDS mappings also exhibit. This property is most significant with respect to FST, since it ensures that a localized fault in the original fault space transforms into a non-localized fault in the transformed fault space.

6 Design of the Non-Linear Layer: Side-Channel Security

Masking of the non-linear substitution boxes (S-Boxes) [20] is a popular countermeasure against differential power analysis (DPA). For ideal security, the mask should be chosen uniformly at random, that is, refreshed after every encryption. This is, however, extremely costly both in terms of area overhead and throughput loss, and is hence rather impractical for lightweight applications targeting resource-constrained devices. As an alternative design strategy, we advocate trading off security with efficiency by refreshing the mask at pre-specified intervals instead of after every

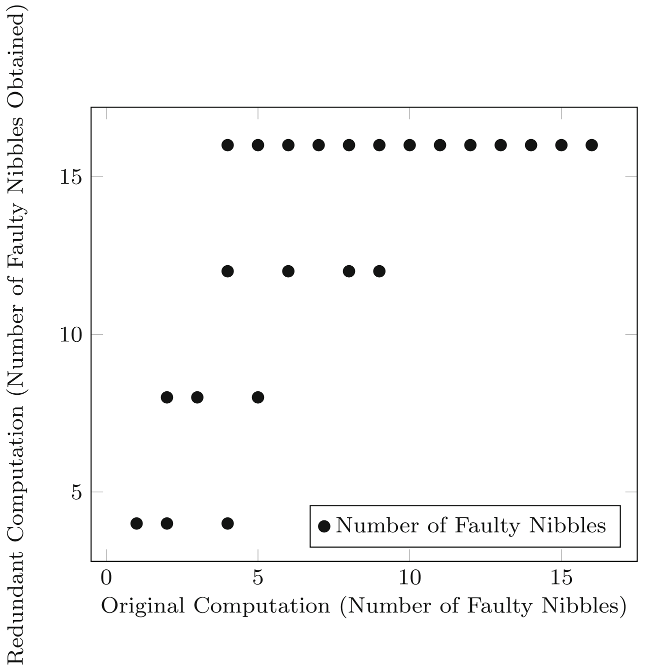


Fig. 4 Effect of fault space transformation on fault distribution: a PRIDE-like block cipher

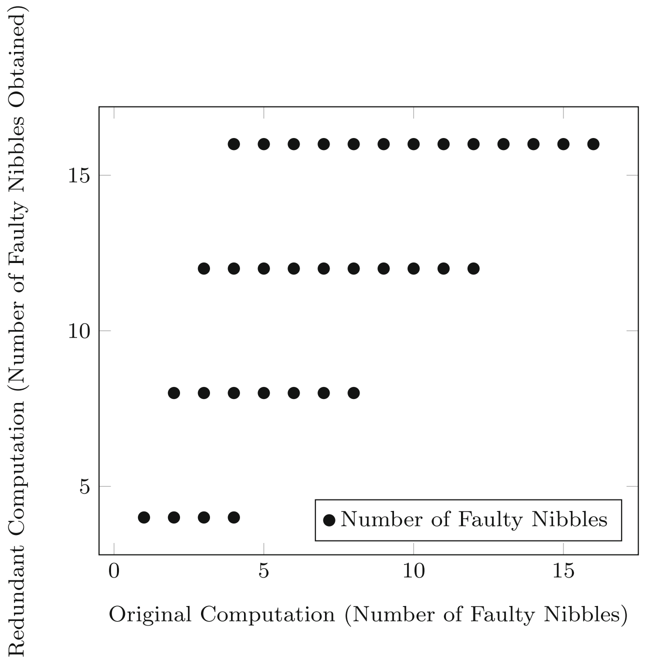


Fig. 5 Effect of fault space transformation on fault distribution: PRESENT-64

encryption. In particular, we focus on serialized block cipher implementations, which is a popular hardware implementation strategy for area-constrained applications. The periodic refresh strategy is further complemented by a careful choice of S-Box for the block cipher. For this, we resort to using *modified transparency order* (MTO) metric [21] to compare between different S-boxes in terms of their resistance against DPA. We establish via simulation studies that there exists a direct correlation between the MTO value of a given S-Box, and the adequate refresh rate for a masked implementation of the same to achieve reasonable side-channel security. In other words, a careful S-Box choice guided by the MTO metric potentially reduces the overhead of periodic mask refresh and improves throughput. Finally, all analysis in this section assumes security against first-order side-channel attacks. While elegant countermeasure techniques exist against higher-order side-channel attacks, they are usually too expensive to implement on resource-constrained devices, and are hence not considered in the following discussion.

6.1 Modified Transparency Order of S-Boxes

The first attempts to quantify the theoretical resistance of S-Boxes against DPA attacks were based on evaluating the signal-to-noise ratio (SNR) of the implementation [53]. The transparency order (TO) [54] was then proposed as a more formal metric focusing on the resilience of S-Boxes against DPA attacks. TO is essentially dependent on the propagation characteristics of the S-Box coordinate functions, and uses a side-channel efficiency metric close to the *standard score measure* introduced in [55]. While the authors of [54] argued that S-Boxes with smaller TO have higher DPA resilience, the definition of TO itself was found to have drawbacks, which were eventually corrected in [21] resulting in a new metric called the MTO.

We begin by recalling the definition of MTO [21] of S-Boxes as a means to evaluate their side-channel resilience. An $n \times m$ S-Box F can be seen as a multi-output Boolean function, namely a function from F_2^n into F_2^m with $m \leq n$. Let $u \in F_2^m$ be a vector whose binary coordinates are all zero except one which is assumed to be at index j, and let $\mathcal{C}_{f_1,f_2}$ denote the *cross-correlation spectrum* between two Boolean functions f_1 and f_2. The jth *component function* of the S-Box F is the single output Boolean function $u \cdot F$. If $F = (F_1, \ldots, F_m)$, then one may note that $u \cdot F = F_j$. We also recall the notion of *cross-correlation spectrum* between two Boolean functions; for $f_1, f_2 \in \mathcal{B}_n$, it is defined for every $\omega \in F_2^n$ as the value $\mathcal{C}_{f_1,f_2}(\omega) = \sum_{x \in F_2^n} (-1)^{f_1(x) \oplus f_2(x \oplus \omega)}$ (note that we have $\mathcal{C}_{f,f}(\omega) = \mathcal{A}_f(\omega)$).

Definition 1 Let F be a balanced $n \times m$ function. Its improved transparency order is the coefficient $\mathsf{TO}(F)$ defined by the following:

$$\mathsf{TO}(F) = \max_{\beta \in \mathbb{F}_2^m} \left(m - \frac{1}{2^{2n} - 2^n} \sum_{a \in \mathbb{F}_2^{n*}} \sum_{j=1}^{m} \left| \sum_{i=1}^{m} (-1)^{\beta_i \oplus \beta_j} \mathcal{C}_{F_i,F_j}(a) \right| \right) . \quad (2)$$

MTO in particular corrects the earlier definition of TO [54] with respect to the assumption that all the coordinates of an S-Box are balanced, which does not apply to the entire space of S-Box functions, such as bent functions. In addition, it also advocates that the cross-correlation terms between S-Box coordinates should be treated as non-zero, unlike in the original definition, where it is treated to be zero. MTO is, to the best of our knowledge, the most comprehensive metric for quantifying DPA resilience of S-Boxes in the cryptographic literature today.

6.2 Masking with Periodic Refresh: Security Versus Lightweight Implementations

Masking of S-Boxes [20] is a popular countermeasure against DPA that aims to randomize the S-Box computations, thereby destroying the correlation of the power leakage with the sensitive data during the execution of the block cipher. Given an S-Box $\mathcal{S}$, an unprotected S-Box implementation would compute $y = \mathcal{S}(x)$ directly. A masked implementation, on the other hand, uses a pair of input and output masks m_x and m_y respectively to compute $y_m = \mathcal{S}(x_m)$, such that $x_m = x \oplus m_x$ and $y_m = y \oplus m_y$ respectively. This ensures that any leakage is independent of the key. For ideal security, the mask should be chosen uniformly at random, that is, refreshed after every encryption. This is, however, extremely costly both in terms of area overhead and throughput loss, and is hence rather impractical for lightweight applications targeting resource-constrained devices. Our aim in this section is to propose a trade-off between security an efficiency by refreshing the mask at pre-specified intervals instead of after every encryption. In particular, we focus on serialized block cipher implementations, which is a popular hardware implementation strategy for area-constrained applications.

We point out here that there are two possible mask refresh strategies for lightweight serialized implementations that could be used in practice. One possibility is to use different masked S-Boxes for each state byte/nibble, and to refresh the masks for each S-Box for each encryption. This is the best possible scenario and affords provable security. The masks for the next encryption could be prepared during the current encryption itself, since a serialized implementation allows sufficient clock cycles during execution for the same. This approach, however, requires the masked S-Boxes for the next round to be stored, leading to an increase in the area footprint of the design. A more lightweight strategy is to use a single masked S-Box for each state byte/nibble, and to refresh the mask periodically to ensure the side-channel leakage does not exceed the safe threshold. This strategy is not provably secure, but is more lightweight while offering reasonable security at the same time. In addition, the security of the system in this case also depends on the choice of the underlying S-Box. Given two S-Boxes, choosing the one supports a lower mask refresh rate without exceeding the safe leakage threshold leads to a secured implementation with a lower cost.

Note that the mask refresh rate critically affects the throughput of any protected block cipher implementation. In particular, a very high refresh rate could catastrophically reduce throughput, making it unsuitable for a large range of real-time applications with reasonably high throughput requirements. High refresh rates also make the implementation vulnerable to fault attacks—an adversary could simply inject faults into the registers used for the mask refresh, instead of the cipher state registers, and bias the mask shares to enhance side-channel leakage. Thus, even though a high mask refresh rate greatly advances resistance against side-channel attacks, it could degrade both performance and security via alternative attack channels. Hence, in our case studies presented subsequently, we explore the trade-off between the refresh rate and the minimum number of samples at which the TVLA evaluation crosses the safe threshold of 4.5. We then choose a refresh rate that ensures that this minimum number of samples is beyond 50,000. Such a refresh rate appears optimal in the sense that (a) it does not lead to a massive decrease in throughput, and (b) the mask refresh operation can be executed at sufficiently unpredictable intervals to prevent an adversary from performing a targeted fault injection in the mask refresh circuit.

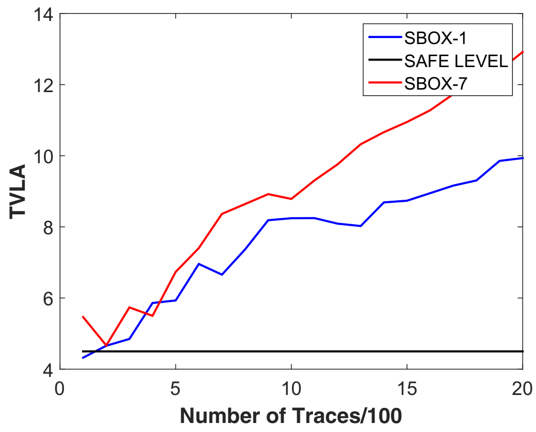
In our case study, we present a side-channel and fault-attack resistant block cipher architecture that makes use of four block RAMs, which are essentially dedicated two-port memory elements available in Xilinx FPGAs, to store the mask shares. The use of Block RAMs not only reduces the overall slice count of the design, but also enhances security against side-channel and fault attacks. We point out that the use of Block RAMs for efficient design of cryptosystems is quite abundant in literature [56, 57]. Indeed, given the fact that nearly all modern FPGAs are equipped with dedicated hard IPs, such as Block RAMs, a judicious utilization of the same for reducing the area footprint of a design targeted primarily for FPGA-based platforms seems quite acceptable. In an actual implementation for deployment in real-world applications, it is preferable to store the mask shares in small tamper-resilient memory units embedded in the target device, so as to prevent corruptions via fault injections.

6.3 Using MTO for Choosing S-Boxes

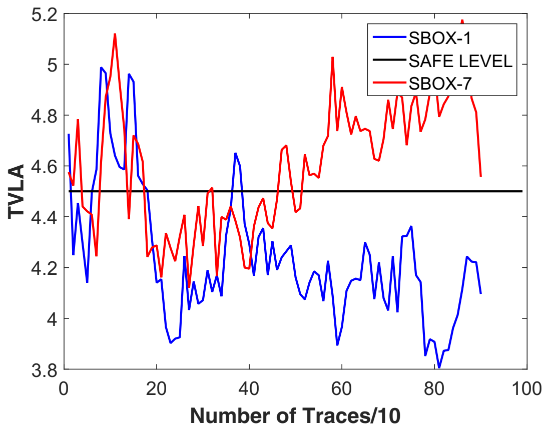
The MTO metric proves very useful in the context of choosing S-Boxes that allow lightweight designs with reasonable security. We explain this by presenting a case-study on 4 × 4 S-Boxes for the block cipher PRINCE listed in [58]. The minimum MTO values of these eight S-Boxes are presented in Table 5. We compare two 4 × 4 S-Boxes out of the eight presented in Table 5—namely S-Box 1 and S-Box 7, having MTO values 1.66 and 2.33 respectively, with respect to their resilience to DPA in the presence of masking with varying refresh rates. Figure 6 summarizes the leakages for both the S-Boxes—in the absence of masking and in the presence of masking with refresh rates of once per

Table 5 Minimum MTO values for the eight PRINCE S-Boxes

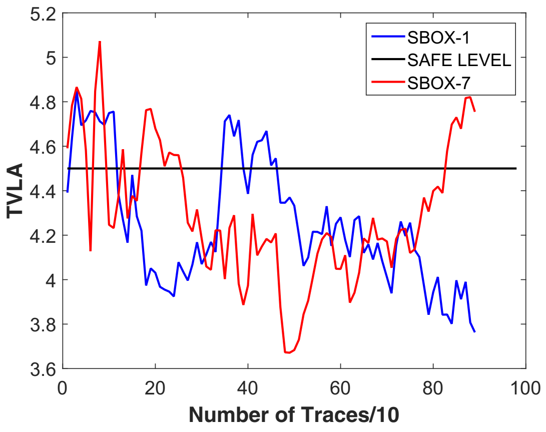
S-Box	Min. MTO
S-Box-1	1.63333
S-Box-2	1.7
S-Box-3	1.66667
S-Box-4	1.56667
S-Box-5	2.16667
S-Box-6	2.1
S-Box-7	2.23333
S-Box-8	2.2



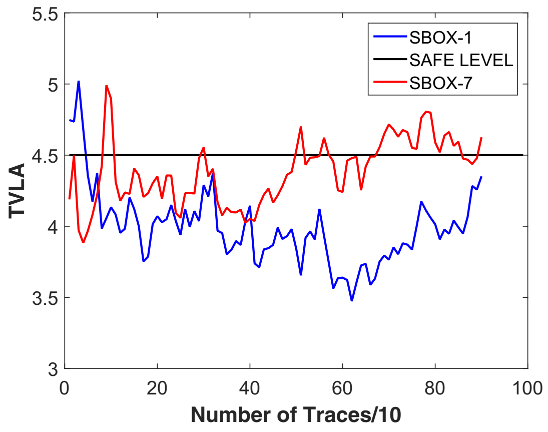
(a) No Masking



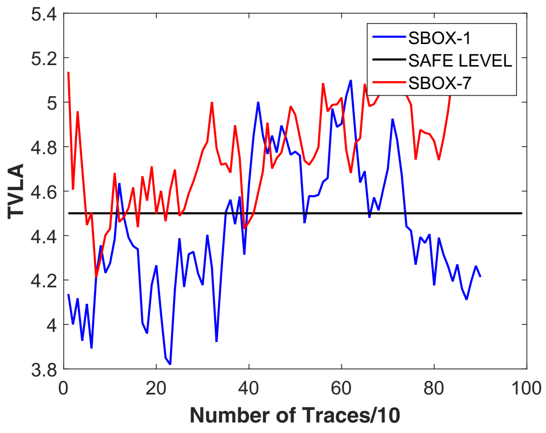
(b) Mask refesh/40 Encryptions



(c) Mask refesh/60 Encryptions



(d) Mask refesh/80 Encryptions



(e) Mask refesh/100 Encryptions

Fig. 6 Comparison of TVLA values for S-Box 1 and S-Box 7 with different mask refresh rates

40, 60, 80, and 100 encryptions respectively. The leakage is measured using the popularly used test vector leakage assessment (TVLA) metric [24]. Quite evidently, in each of the cases, S-Box 7 exhibits TVLA leakage greater than the safe threshold of 4.5 in a fewer number of traces than S-Box 1 (excluding the initial irregularities arising from unexploitable forms of leakage). *Thus, given a set of S-Boxes with widely varying MTO values, choosing the S-Box with lesser TO leads to a secured masked implementation with lesser cost.*

6.4 Qualitative Comparison with Fresh Re-keying

Fresh re-keying is an elegant technique that can be used to thwart a number of implementation-based attacks, including DPA and fault attacks. A nice exposition on the advantages of fresh re-keying in RFID technology is presented in [59]. The authors of [59] propose a fresh re-keying scheme that is especially suited for challenge-response protocols such as used to authenticate tags. A qualitative comparison reveals certain similarities between masking with periodic refresh and fresh re-keying: in both cases, the implementation uses an additional function acting as a source of randomness to prevent an adversary from collecting sufficient number of correlated leakage traces from a given cipher implementation. For fresh re-keying this is the function used to refresh the key, while in masking, this is the function used to generate the fresh mask shares. Additionally, the refresh rate in each implementation directly influences security: a refresh for every encryption renders certain forms of attacks impossible to mount, while a lower refresh rate results in a trade-off between security and efficiency. We believe that for our approach provides an alternative to the fresh re-keying mechanism, both in terms of security and efficiency, especially in applications other than those based on challenge-response protocols, where fresh re-keying may not be viable.

7 Strenghtening Side-Channel Resistance: Lightweight Shuffling Across Rounds

This section presents the idea of shuffling across multiple rounds as a cipher-dependent countermeasure strategy to prevent side-channel attacks on lightweight serialized implementations of block cipher algorithms. In a shuffled implementation, any key-dependent operation is deranged in time, thereby increasing the uncertainty of the adversary as to when a specific key-dependent operation occurs in time. This in turn reduces the SNR of the implementation, thereby increasing security against side channels [22]. Standard shuffling practices often advocate shuffling to be restricted within the operations in a single round. This often does not afford sufficient security for shuffling to be used as a standalone countermeasure. In this section, we present details of how shuffling may be done across multiple rounds of a cryptographic algorithm, and how this depends on the nature of the underlying block cipher. Our analysis is generic enough to be applicable to a wide range of block ciphers.

7.1 SNR and Shuffling

Suppose that during the execution of a cryptographic algorithm, a side-channel adversary targets a specific operation, which depends on a specific subpart of the secret key K and occurs at time t_c in an execution of the cryptographic algorithm. Let $\hat{P}$ be the leakage from the device at time $\hat{t}_c$, which is composed of two component leakages—P due to the actual processing of the target intermediate result (also referred to as the *signal*), and N due to device and algorithmic noise. Let the hypothetical leakage corresponding to the correct sub-key guess corresponding to the target operation be H. Also, let $\rho(H, \hat{P})$ and $\rho(H, P)$ denote the correlation coefficient between the hypothetical power consumption and the total and signal (actual) leakages respectively.

Then, as per the formulation stated in [22], we have the maximum correlation as follows:

$$\rho_{\max}(H, \hat{P}) = \frac{\rho(H, P)}{\sqrt{1 + \frac{1}{\mathrm{SNR}}}} \tag{3}$$

Thus, quite evidently, lower the SNR, the easier it is for an adversary to attack the given cryptographic implementation.

Finally, the number of samples S required to mount a successful SCA attack is related to the maximum correlation $\rho(H, \hat{P})$ as follows:

$$S = 3 + 8\left(\frac{Z_\alpha}{\ln\left(\frac{1+\rho(H,\hat{P})}{1-\rho(H,\hat{P})}\right)}\right)^2 \tag{4}$$

where Z_α denotes the distance between the sampling distributions with minimum and maximum correlation coefficients and is a device dependent parameter [22].

In a shuffled implementation, any key-dependent operation is deranged in time. There now exists a finite probability $\hat{p}$ that the target operation occurs at a particular time t_c. In such a scenario, the expression for $\rho(H, \hat{P})$ takes the form:

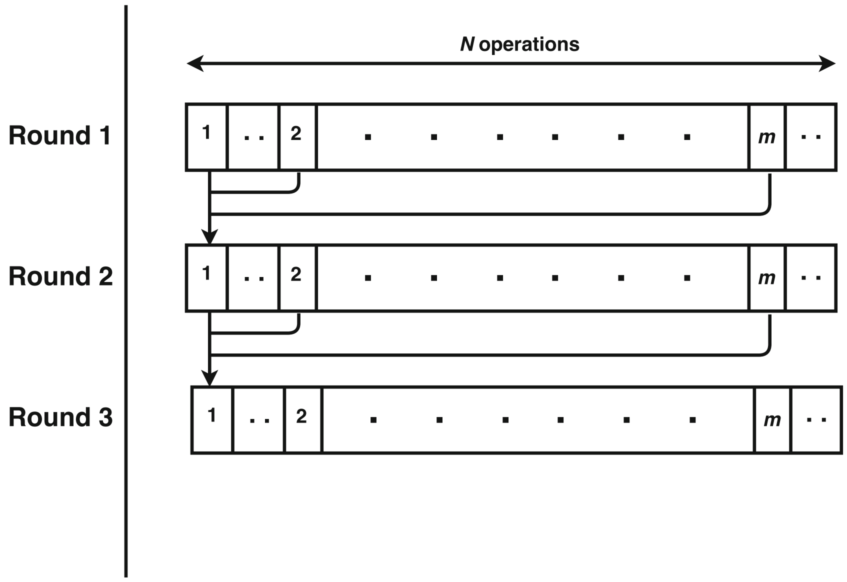
$$\rho_{\max}(H, \hat{P}) = \frac{\rho(H, P)}{\sqrt{1 + \frac{1}{\mathrm{SNR}}}} \times \sqrt{\hat{p}} \times \sqrt{\frac{\mathrm{Var}(P)}{\mathrm{Var}(\hat{P})}} \tag{5}$$

Quite evidently, greater the extent of derangement due to shuffling, lower is the value of $\hat{p}$ and the more difficult it is for the adversary to mount an SCA, such as integrated DPA on the implementation.

7.2 Block Cipher Structure-Dependent Shuffling Across Multiple Rounds

We introduce the concept of generalized shuffling across multiple rounds of a cryptographic algorithm. We assume the following parameters for the structure of the underlying

Fig. 7 The inter-round dependencies : an illustration for three rounds



block cipher, which in turn impact the prospects of the multi-round shuffling algorithm:

- The cipher algorithm consists of R rounds
- Each round r (where $1 \leq r \leq R$) consists of exactly N key-dependent operations that may be targeted by an adversary (such as S-Box computations)
- Each potential target operation in round $r+1$ depends on the output of exactly m operations from round r

We refer to m as the *degree of diffusion* for the block cipher (Fig. 7).

7.3 The Derangement Space

We now analyze mathematically the impact of shuffling the operations across multiple rounds on the derangement space (the number of clock cycles across, which a sensitive operation may be permuted) for each operation. Let the operations in a serialized implementation of the target block cipher be shuffled across k rounds, for $1 \leq k \leq N$, and let r be any arbitrary round in this shuffling range. For maximum diffusion across rounds, it is logical to assume that the computation of each potential target operation in round r depends on exactly $\min(m^i, N)$ operations in round $r-i-1$. Thus, for $l = \log_m N$, any operation in round r depends on all operations in rounds prior to $r-l-1$, and affects every operation in rounds succeeding $r+l$. Such absolute dependencies therefore cannot contribute to the uncertainty due to shuffling.

Then, any shuffling algorithm for k rounds must satisfy the following categories of constraints, as depicted pictorially in Fig. 8. Category 1 and category 4 enumerate complete dependencies, while categories 2 and 3 enumerate partial dependencies across rounds.

1. Category 1: Any operation in round r depends on each of the N operations computed in rounds $1, \cdots, r-l$ (assuming $r > l$)
2. Category 2: Any operation in round r depends on exactly $m^{l-1}, m^{l-2}, \cdots, m$ operations computed in rounds $r-l+1, r-l+2 \cdots, r-1$
3. Category 3: Any operation in round r affects the output of exactly $m, m^2, \cdots, m^{l-1}$ operations computed in rounds $r+1, r+2, \cdots, r+l-1$
4. Category 4: Any operation in round r affects the output of all N operations in rounds $r+l, \cdots, k$ (assuming $k \geq r+l$)

Keeping the aforementioned dependencies in mind, we argue that an arbitrarily large value of k is not desirable for the following reasons:

- A very large value of k makes generating permutations of size Nk a computationally intensive task, and sacrifices the very lightweightness that makes shuffling an attractive countermeasure choice.
- A very large value of k brings a large number of dependencies of categories 1 and 4, which are rather deterministic from the point of view of the adversary, and do not contribute to the overall randomness of the permutation.

In particular, we propose choosing values of k such that $k \leq l+1$. This ensures that for all values of r where $1 \leq r \leq k$, we have $r \leq l+1$ and $k \leq r+l$.

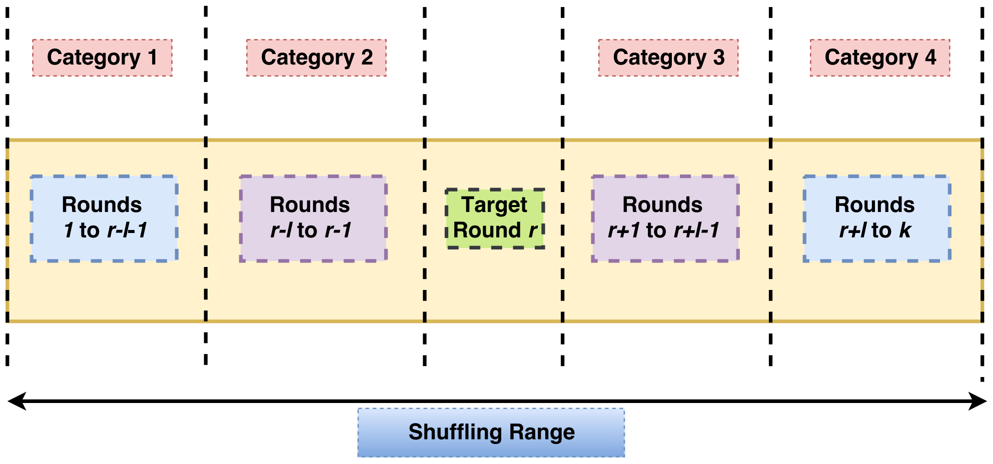


Fig. 8 Shuffling Across k rounds : the constraints for a random round r

Now, let D_r be the maximum possible range in time over which any operation in round r may be effectively deranged. It is easy to see that for shuffling across k rounds, the derangement space D_r may be lower bounded as follows:

$$\begin{aligned} D_r &= \mathrm{Nk} - N(r-l-1) - \frac{m^l - m}{m-1} - \frac{m^{k-r+1}-m}{m-1} \\ &= \mathrm{Nk} - \left(\frac{m^r - m}{m-1}\right) - \left(\frac{m^{k-r+1}-m}{m-1}\right) \\ &= \mathrm{Nk} - 2\left(\frac{m^l - m}{m-1}\right) \\ &\geq \mathrm{Nk} - 2\left(m + \cdots + m^{l-1}\right) \end{aligned} \tag{6}$$

We point out that while multi-round shuffling seems to be a generic countermeasure idea, a precise implementation of the same is heavily dependent on the structure of the target cipher. For example, for block ciphers with a substitution-permutation network based structure (such as AES or PRESENT), the maximum number of rounds to which shuffling may be extended is given by $\log_m N$, where N is the number of operations in each round and m is the degree of diffusion.

In addition, the derangement space (and consequently the degree of security afforded) is a also a function of the N and m.

Note that for one-round shuffling, the value of D_r is N, which follows from the fact that only the operations within one round are being shuffled. For $k = 2$, we have $D_r = 2N - m$.

7.4 Security of *k*-round Shuffling Against Side-Channel Attacks

We now examine the impact of shuffling across multiple rounds on the side-channel resistance of the target block cipher. Recall from Eq. 5 that the maximum correlation $\rho(H, \hat{P})$ between the hypothetical and the total power consumptions (H and $\hat{P}$ respectively) depends on the derangement probability $\hat{p}$ for the given target operation. Assuming that our shuffling algorithm ensures that a given operation in round r is permuted uniformly at random across its entire derangement space D_r, the derangement probability $\hat{p}$ is given by the following:

$$\hat{p}_{k\text{–round shuffling}} = \frac{1}{Nk - \left(\frac{m^r + m^{k-r+1} - 2m}{m-1}\right)} \tag{7}$$

Our next step is to relate the increase in derangement space due to multi-round shuffling with the corresponding increase the number of samples required to attack a block cipher implementation, as compared to an unprotected version of the same. This analysis follows from the standard relation of the SNR of an implementation with the number of attack samples required (refer [22] for more details):

$$\begin{aligned} \frac{S_{k\text{–round shuffling}}}{S_{\text{unprotected}}} &\approx \left(\frac{\ln\left(\frac{1+\rho_{\text{unprotected}}}{1-\rho_{\text{unprotected}}}\right)}{\ln\left(\frac{1+\rho_{k\text{–round shuffling}}}{1-\rho_{k\text{–round shuffling}}}\right)}\right) \\ &\approx \left(\frac{\rho_{\text{unprotected}}}{\rho_{k\text{–round shuffling}}} \cdot \frac{1-\rho_{k\text{–round shuffling}}}{1-\rho_{\text{unprotected}}}\right) \\ &\approx \left(\frac{\rho_{\text{unprotected}}}{\rho_{k\text{–round shuffling}}}\right) \\ &\geq \left(Nk - \left(\frac{m^r + m^{k-r+1} - 2m}{m-1}\right)\right) \end{aligned} \tag{8}$$

Thus, given a block cipher parameterized by (N, m), our aforementioned analysis allows us to evaluate a lower bound on the side-channel security afforded by shuffling across

k rounds in terms of the derangement space afforded by multi-round shuffling. For example, for the block cipher PRESENT with $N = 16$ and $m = 4$, shuffling across $k = 2$ rounds is expected to have a derangement space around 1.75 times larger than traditional shuffling within a single round. This in turn affords greater security against side-channel attacks. Note that both the security and the implementation overhead for multi-round shuffling is heavily dependent on factors, such as N and m, and hence, on the structure of the underlying block cipher.

7.5 Unification of Masking and Multi-Round Shuffling

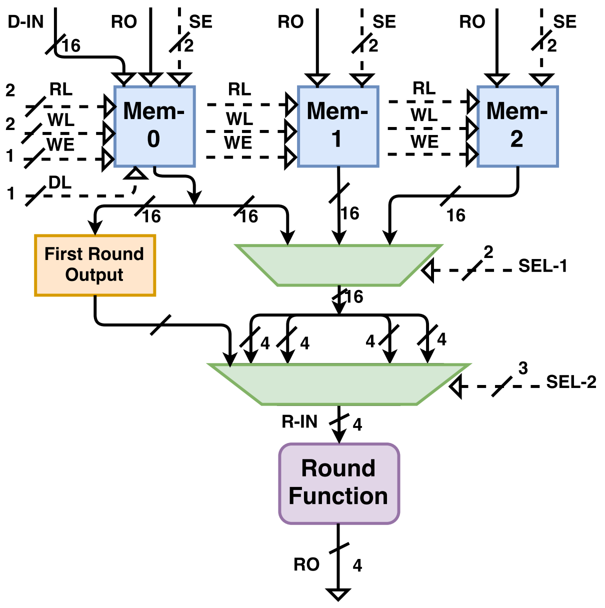
We conclude our proposals for lightweight design-for-security against die channel attacks by advocating a combination of masking with periodic refresh, and shuffling across rounds. From the point of view of security, it is enough to have only masking, with a refresh rate of once per encryption, which makes the system provably secure against side-channel attacks such as DPA. However, in resource-constrained environments, this is too expensive and often impractical to achieve without compromising hugely on the throughput. Hence, any practical lightweight system can only afford masking with a limited mask refresh rate, which in turn implies compromising on security. Such a system can, however, be *fortified further* against side-channel threats by the incorporation of multi-round shuffling as proposed above. This leads to systems with strong security and, at the same time, lesser cost of implementation than provably secure masking.

8 Case-Study 3: Combining Masking with Refresh and Two-Round Shuffling for PRESENT

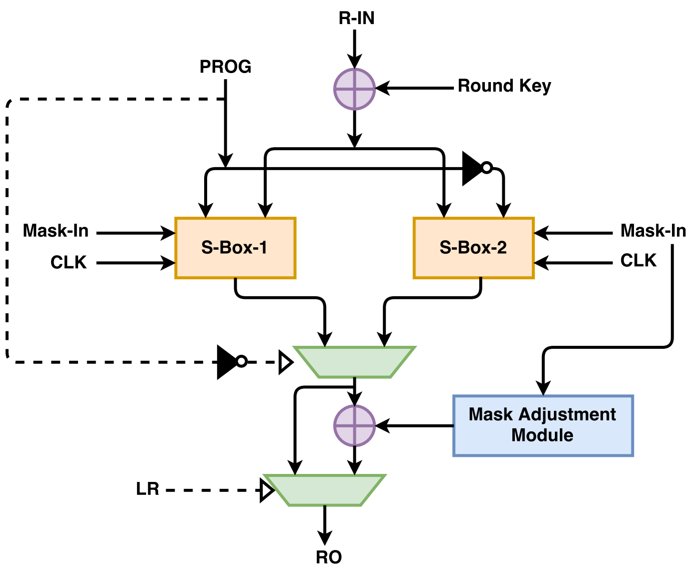
In this section, we present a case study on a PRESENT-64 where we combine both the aforementioned strategies of masking with periodic refresh and shuffling across two rounds in order to resist side-channel attacks. Our aim here is to demonstrate that while neither of these lightweight strategies can serve as stand-alone countermeasures, their combination results in a design that has sufficient security against side-channel attacks for IoT-based applications. In addition, the design is more lightweight in terms of area footprint as compared to the most efficient two-share threshold implementation of PRESENT [60] reported in the existing literature.

8.1 Implementation Details

In this section, we present details of our implementation of PRESENT that combines both masking with periodic refresh and shuffling across two rounds. Figure 9 illustrates the basic serialized architecture of our design that processes



(a) The Overall Architecture



(b) The Round Function

Fig. 9 Combined masking and two-round shuffling on PRESENT-80

one nibble per clock cycle. We make use of Block RAMS—dedicated two-port memory elements in Xilinx FPGAs—as storage elements for the block cipher state. Three such memory blocks are maintained in the architecture in order to accommodate two-round shuffling. While one of these stores the content of the previous round, the other two are used to update the state nibbles of the current and next round in random shuffled order. The cross-dependencies across nibbles in consecutive rounds (induced by the bit permutation layer) are handled using a combination of multiplexers. The masked S-Box for the first two rounds is prepared initially before the start of the encryption, while the masked S-Box for each round $r + 1$, where $r \geq 1$, is programmed in parallel to the execution of round r (the control signal PROG is used to handle the parallel S-Box programming). Thus, the mask gets refreshed after every two rounds of execution.

8.1.1 Area Overhead Results

The resource overhead for our implementation of PRESENT that combines masking with two-round shuffling is presented in Table 6. The target platform is a Xilinx xc5vlx50 FPGA. Table 6 also compares the overhead of our implementation with that of existing threshold implementations for PRESENT against first-order side-channel attacks. The comparison results clearly demonstrate that our implementation is more lightweight than all existing threshold implementations of PRESENT in terms of both LUT and flip-flop requirements. While our design uses four additional BRAMs, we would like to point out that Table 6 presents the area overheads only for the encryption modules of each design; all TI implementations reported here require additional area for pseudo-random number generation and other pre-computation steps such as in [62]. Hence, the use of four additional BRAMs does not make our design more expensive that TI implementations.

8.1.2 Power Consumption

With respect to power consumption, we calculated the average and maximum power consumed by our side-channel resistant implementation, which were found to be 225.6 and 226.2 mW, respectively. This is around 12% more energy consumption as compared to an unprotected serialized implementation of PRESENT, that consumes around 205.7 mW of power on an average, and a maximum of 210.1 mW. The consumed power was measured using the voltage drop across a 1-ohm shunt register on a SASEBO GII board.

8.2 Leakage Analysis

In this section, we present an analysis of the security of our proposed countermeasure strategy against side-channel attacks. The analysis is presented for each of the four configurations mentioned above using the popularly used test vector leakage assessment (TVLA) metric [24]. Figure 10 presents the TVLA leakage of the four configurations across 50,000 power traces collected on a SASEBO GII board. A comparison of the maximum TVLA progression with the number of collected traces for each configuration is presented in Fig. 11. We briefly comment about the leakage for each configuration below (the area overhead for each configuration can be found in Table 7):

1. *Unprotected Implementation*: The unprotected implementation of PRESENT obviously suffers from the maximum leakage due to the absence of any countermeasures against side-channel attacks.
2. *Masking with Refresh*: The second maximum leakage rate comes from the implementation with only masking. This is probably a bit surprising, but maybe explained as follows. The mask share is refreshed only once in 32 clock cycles, and the same masked S-Box is used

Table 6 Comparison of implementation overhead

Implementation (encryption module)	Slice count (look-up tables)	Slice count (registers)	Block RAM	Maximum frequency	Target FPGA	Synthesis tool
Threshold (first order) [61]	641	384	0	218 MHz	Xilinx Spartan-6 LX75	Xilinx XST
Threshold (first order) [62]	808	384	0	207 MHz		
Threshold (second order) [62]	2245	1680	0	204 MHz		
Threshold (first order) [63]	1720	722	0	112 MHz		
Threshold (first order) [60]	742	362	0	490 MHz	Xilinx Virtex-5 xc5vlx50	
Shuffling + masking (this paper)	570	308	4	105 MHz	Xilinx xc5vlx50	

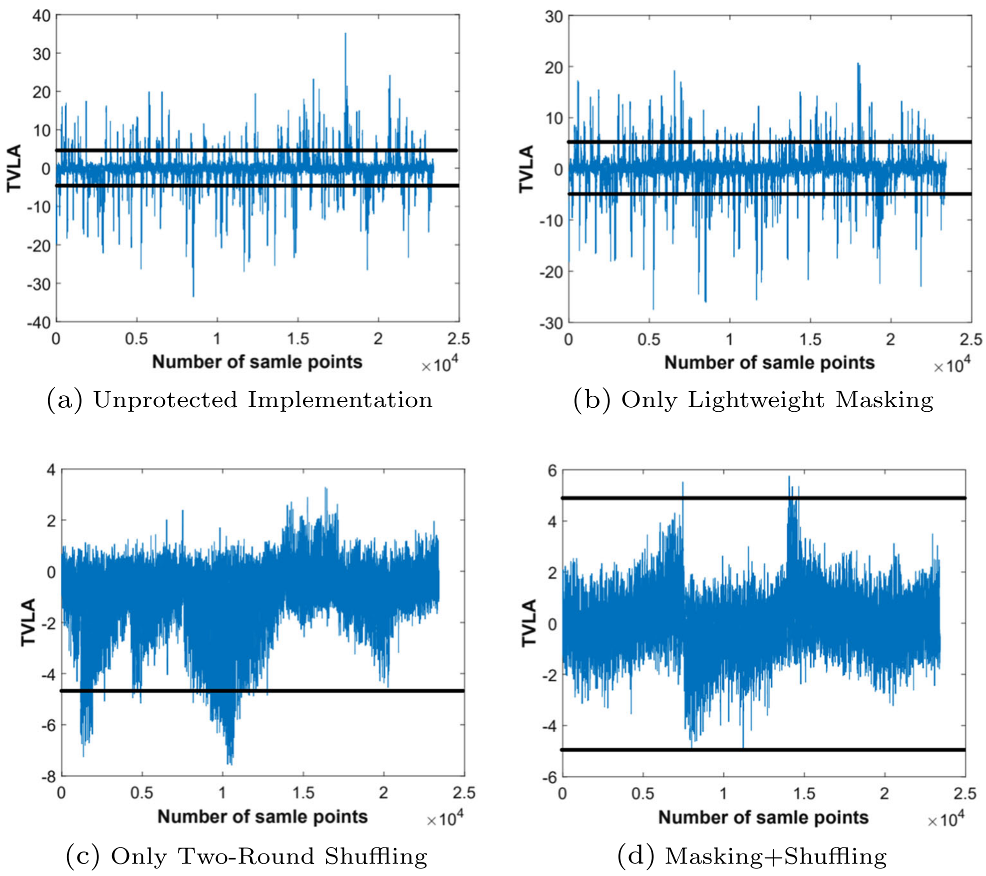


Fig. 10 TVLA leakage on 50,000 traces for different implementations using PRESENT S-Box

across two rounds of computation. This is thus a rather lightweight form of masking that compromises most of the requirements of normal masking, such as in threshold implementations [60, 64]. Such a masking strategy is therefore not meant to serve as a stand-alone countermeasure against side-channel attacks. However, in conjunction with other low-cost countermeasures such as shuffling, it improves the security of the implementation.

3. *Two-Round Shuffling*: Two-round shuffling has a much lower leakage as compared to the unprotected and masking-only implementations. It is, again, a lightweight countermeasure that is stronger than one-round shuffling but not secure enough to serve as a stand-alone countermeasure. This is not a surprising observation since, as per our analysis in Section 7.1 shuffling increases the number by samples required to break the system by around 28 times, which is a linear increase in security.
4. *Masking + Shuffling*: This implementation is the most secure and barely crosses the safe threshold over 50,000 power traces. Thus, quite evidently, combining two lightweight countermeasure techniques, none of which are sufficiently secure in isolation, we arrive at a design that is both lightweight and more secure than of the aforementioned techniques in isolation.

For a more exact estimation of the actual strength of our countermeasure, we state that the TVLA leakage from the design using the PRINCE S-Box is found to exceed the threshold of 4.5 at around 150,000 traces.

9 Case-Study 4: Combining Side-Channel Analysis with Fault Resistance for a PRESENT-like Block Cipher

We present a final case study where we assimilate our proposed design strategies for protection against side-channel and fault attacks. In particular, we make two main alterations/additions to the implementation of PRESENT reported in Section 8: (a) we incorporate a spatial redundancy-based FST in the design according to strategies discussed in Section 2, and (b) we implement the design principle for S-Boxes proposed in Section 6. In particular, we substitute the PRESENT S-Box with the candidate 4×4 S-Box for PRINCE listed in [58] that has the least MTO among all such candidate S-Boxes (see S-Box-4 in Table 5 with a mini-

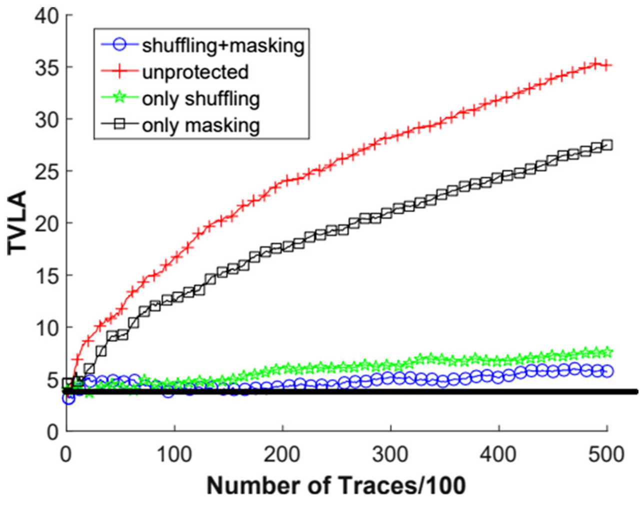


Fig. 11 Progression of TVLA with number of power traces: implementations using PRESENT S-Box

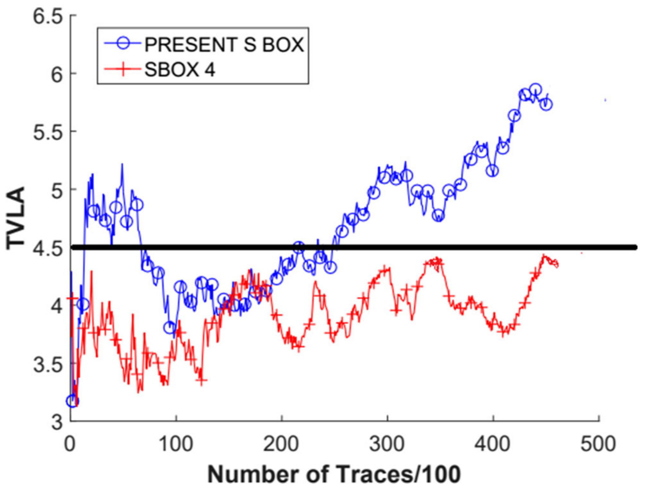


Fig. 12 TVLA leakage on 50,000 traces: combined side-channel and fault attack resistance using PRINCE S-Box

mum MTO value of 1.56667). We also present an equivalent case study for the block cipher GIFT proposed in [19], where we combine spatial redundancy-based FST using the bit permutation layer of GIFT with a refreshing-based masked implementation of the PRINCE S-Box and shuffling of S-Box operations across rounds. Since the TVLA leakage for the protected GIFT implementation is found to be nearly identical to that of the PRESENT implementation, we only report the area overhead in this case. The TVLA leakage for the PRESENT-like implementation is presented in Fig. 12 while the architecture for the protected GIFT-64 implementation is presented in Fig. 13.

9.1 Area Overhead of Our Combined Strategy

The resource overhead for our design is presented in Table 8, and is compared against the combined countermeasure presented by Cnudde et al. in [64], which is the only other combined countermeasure strategy targeting PRESENT for FPGA-based implementations, to the best of our knowledge. The countermeasure of Cnudde et al. uses a threshold implementation to counter side-channel attacks, and combines the same with private circuits (PC-II) to resist fault attacks. Table 8 presents an overall comparison of their implementation with ours, while Table 6 presents a direct comparison of the overhead specific to side-channel resistance. Quite clearly, our strategy leads to a more lightweight implementation; albeit, at the cost of a trade-off on security, as described in the following section. Note that while the authors of [64] have recently presented certain extensions to their original proposal in [65], the results in [65] entirely target ASIC platforms, and hence do not provide scope for fair comparison. We therefore restrict our comparison studies to the earlier version of their paper reporting FPGA implementations. We also note that the relatively higher overhead for the GIFT-64 implementation may be attributed to the fact that we completely avoid the use of block RAMs for this design. This also provides an estimate of the relative costs of our approach for designs with and without block RAMs.

9.2 Security of Our Combined Strategy

Figure 12 summarizes the TVLA leakage over 50,000 power traces for two designs—one with the PRESENT S-Box and the other with the PRINCE S-Box. Both the designs combine FST with lightweight masking and two-round shuffling. Note that the use of an S-Box with lower MTO significantly reduces the TVLA leakage. In particular, while the TVLA leakage from the design using the PRESENT S-Box exceeds the threshold of 4.5

Table 7 Area overhead for different implementations

Implementation (encryption module)	Slice count (look-up tables)	Slice count (registers)	Block RAM	Maximum frequency	Target FPGA	Synthesis tool
Unprotected implementation	215	123	0	205 MHz	Xilinx xc5vlx50	Xilinx XST
Only lightweight masking	428	235	1	112 MHz		
Only two-round shuffling	354	196	3	145 MHz		
Masking + shuffling	570	308	4	105 MHz		

Table 8 Implementation overhead: combined side-channel and fault attack resistance

Implementation (encryption module)	Slice count (look-up tables)	Slice count (registers)	Block RAM	Maximum frequency	Target FPGA	Synthesis tool
TI+PC-II [64]	10341	1292	0	Not reported	Xilinx Virtex-II Pro	Xilinx XST
Shuffling + masking + FST (PRESENT-80)	655	344	4	65.7 MHz	Xilinx xc5vlx50	
Shuffling + masking + FST (GIFT-64)	1244	1081	0	74.9 MHz		

within 50,000 traces, the leakage from the design using the PRINCE S-Box is within the secure threshold over 50,000 traces. This establishes the validity of our proposed design choice for S-Boxes. In summary, as compared to the countermeasure strategy of Cnudde et al., our proposal is more suited to applications targeting highly resource-constrained environments, and requiring reasonable security guarantees against both side-channel and fault attacks.

9.3 Comparison with ParTI [66]

We present a comparative study of our proposed combined countermeasure strategy with ParTI [66]—a recent study that protects the block cipher LED against side-channel and fault attacks. Note that the comparison here is purely of design principles and general overheads incurred, as opposed to concrete implementation-based comparison, since the target block cipher for ParTI is LED, while the target block ciphers for our approach are PRESENT-80 and GIFT-64. ParTI essentially combines a threshold implementation of LED with error-detection codes, while our strategy combines a combined masking and shuffling-based lightweight implementation of PRESENT/GIFT with bit permutation-based FST. In terms of design principle, we point out certain advantages of our strategy over that of ParTI, especially with respect to IoT applications:

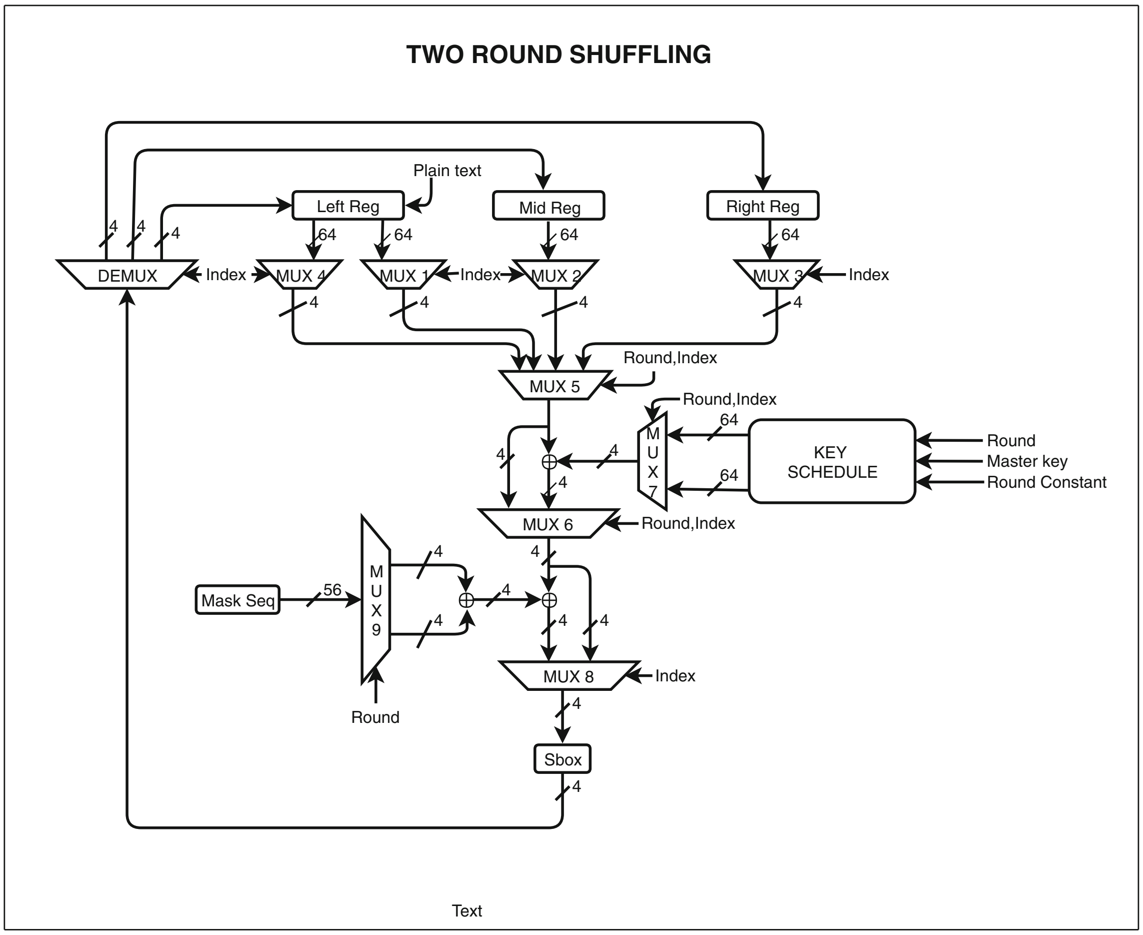


Fig. 13 Architecture for protected implementation of GIFT-64

Table 9 Comparison of proposed method with other combined countermeasure strategies

Countermeasure technique	Target cipher	Additional overhead (in times)
ParTI [66]	LED-64	19.92x
Shuffling + masking + FST	PRESENT-80	3.12x
	GIFT-64	6.11x

1. As already discussed, our strategy of combining lightweight masking with two-round shuffling leads to more resource-efficient side-channel resistant designs with reasonable security margins as compared to threshold implementations (see, for example, Table 6 for a comparison of our implementation with existing threshold implementations for PRESENT). In many IoT applications where area constraints are paramount, our technique provides a more suitable alternative.
2. ParTI uses error-detection codes such as parity, and hence requires additional hardware for predictors and error-detection modules. Our strategy, on the other hand, uses FST which, particularly in the case of PRESENT, is zero-cost in hardware since it only uses bit permutations.

Table 9 compares the implementation overheads of ParTI and our proposed implementations of PRESENT and GIFT. Since the implementations target ASIC and FPGA platforms respectively, we highlight the relative increase in overhead for either design as compared to baseline unprotected implementations of LED and PRESENT, respectively. The unprotected and protected implementations in either case are reported for the same ASIC technology/FPGA platform. In particular, the baseline implementation overhead for unprotected LED is obtained from [67], which uses the same 0.18 μm technology as used in ParTI. Observe that the overhead in case of ParTI is 19.92 times that for the baseline implementation, while that for our implementations of PRESENT and GIFT are 3.12 times and 6.11 times, respectively (once again, the difference may be attributed to the fact hat the GIFT implementation does not use block RAMs). The difference in overhead due to our approach with that incurred in the ParTI approach can be principally attributed to the use of FST in our proposed approach, which leads to more lightweight designs.

10 Conclusion

In this paper, we addressed the problem of design-for-security methodologies to construct lightweight block ciphers for IoT applications with combined protection against both side-channel and fault attacks. Our target platforms were FPGAs, which are extremely suitable for IoT owing to their wide range of reconfigurable properties. We proposed three main design principles—the first pertaining to fault attack protection, and the remaining pertaining to side-channel protection. With respect to fault attacks, we proposed recursive design strategies for MDS linear layers with lightweight roots. Such linear layers serve the dual purpose of providing diffusion for resistance against classical cryptanalysis, and ensuring FST for countering both DFA- and DFIA-like fault attacks. In the context of side-channel protection, we proposed a combination of two lightweight strategies—masking with periodic refresh and shuffling across multiple rounds. Finally, we presented a case study on a PRESENT-like block cipher that combined all our design choices into a single lightweight implementation with holistic resistance against both side-channel and fault attacks. Our design requires around 15% lower resources as compared to the best threshold implementations of PRESENT in the literature.

Acknowledgments The authors would like to thank the anonymous reviewers of Journal of Hardware and Systems Security for their helpful comments and suggestions.

Funding Information This study is supported by to the Intel Corporation, USA for partial funding of the work under the project “LightCrypto: Ultra-Light-weight Robust Crypto-Architectures for Performance and Energy.”

References

1. Kocher P, Jaffe J, Jun J (1999) Differential power analysis. In: Advances in cryptology, CRYPTO’99. Springer, pp 388–397
2. Tunstall M, Mukhopadhyay D, Ali S (2011) Differential fault analysis of the advanced encryption standard using a single fault. In: Information security theory and practice. Security and privacy of mobile devices in wireless communication. Springer, pp 224–233
3. Benini L, Macii A, Macii E, Omerbegovic E, Pro F, Poncino M (2003) Energy-aware design techniques for differential power analysis protection. In: Proceedings of the 40th design automation conference, DAC 2003, Anaheim, CA, USA June 2-6, 2003, pp 36–41
4. Standaert F-X, Peeters E, Quisquater J-J (2005) On the masking countermeasure and higher-order power analysis attacks. In: International conference on information technology: coding and computing, 2005. ITCC 2005, vol 1. IEEE, pp 562–567
5. Moradi A, Poschmann A (2010) Lightweight cryptography and dpa countermeasures: a survey. In: Financial cryptography and data security. Springer, pp 68–79

6. Guo X, Karri R (2013) Recomputing with permuted operands: a concurrent error detection approach. IEEE Trans Comput-Aided Design Integ Circuits Syst 32(10):1595–1608
7. Tupsamudre H, Bisht S, Mukhopadhyay D (2014) Destroying fault invariant with randomization. In: Cryptographic hardware and embedded systems–CHES 2014. Springer, pp 93–111
8. Thuy Ngo X, Bhasin S, Danger J-L, Guilley S, Najm Z (2015) Linear complementary dual code improvement to strengthen encoded circuit against hardware trojan horses. In: 2015 IEEE international symposium on hardware oriented security and trust (HOST). IEEE, pp 82–87
9. Ghalaty N, Yuce B, Taha M, Schaumont P (2014) Differential fault intensity analysis. In: Proceedings 2014 workshop on fault diagnosis and tolerance in cryptography (FDTC), vol 2014. IEEE, pp 49–58
10. Patranabis S, Chakraborty A, Nguyen PH, Mukhopadhyay D (2015) A biased fault attack on the time redundancy countermeasure for AES. In: Constructive side-channel analysis and secure design. Springer, pp 189–203
11. (2001). FIPS PUB 197. Advanced Encryption Standard. National Institute of Standards and Technology
12. Bogdanov A, Knudsen LR, Leander G, Paar C, Poschmann A, Robshaw MJB, Seurin Y, Vikkelsoe C (2007) PRESENT: an ultra-lightweight block cipher. In: Cryptographic hardware and embedded systems - CHES 2007, 9th international workshop, Vienna, Austria, September 10-13, 2007, Proceedings, pp 450–466
13. Beaulieu R, Shors D, Smith J, Treatman-Clark S, Weeks B, Wingers L (2015) The SIMON and SPECK lightweight block ciphers. In: Proceedings of the 52nd annual design automation conference, San Francisco, CA, USA, June 7-11, 2015, pp 175: 1–175:6
14. Karpovsky M, Kulikowski KJ, Taubin A (2004) Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard. In: 2004 international conference on dependable systems and networks. IEEE, pp 93–101
15. Regazzoni F, Eisenbarth T, Breveglieri L, Ienne P, Koren I (2008) Can knowledge regarding the presence of countermeasures against fault attacks simplify power attacks on cryptographic devices? In: IEEE international symposium on defect and fault tolerance of VLSI systems, 2008. DFTVS'08. IEEE, pp 202–210
16. Patranabis S, Chakraborty A, Mukhopadhyay D, Chakrabarti PP (2017) Fault space transformation: a generic approach to counter differential fault analysis and differential fault intensity analysis on aes-like block ciphers. IEEE Trans Inf Forensics Secur 12(5):1092–1102
17. Albrecht MR, Driessen B, Kavun EB, Leander G, Paar C, Yalçın T (2014) Block ciphers–focus on the linear layer (feat. pride). In: Advances in cryptology–CRYPTO 2014. Springer, pp 57–76
18. Bogdanov A, Knudsen LR, Leander G, Paar C, Poschmann A, Robshaw MJB, Seurin Y, Vikkelsoe C (2007) PRESENT: an ultra-lightweight block cipher. Springer, Berlin
19. Banik S, Pandey SK, Peyrin T, Sasaki Y, Sim SM, Todo Y (2017) GIFT: a small present - towards reaching the limit of lightweight encryption. In: CHES, volume 10529 of lecture notes in computer science. Springer, pp 321–345
20. Maghrebi H, Danger J-L, Flament F, Guilley S, Sauvage L (2009) Evaluation of countermeasure implementations based on Boolean masking to thwart side-channel attacks. In: 2009 3rd international conference on signals, circuits and systems (SCS). IEEE, pp 1–6
21. Chakraborty K, Sarkar S, Maitra S, Mazumdar B, Mukhopadhyay D, Prouff E (2017) Redefining the transparency order. Des Codes Crypt 82(1-2):95–115
22. Mangard S (2004) Hardware countermeasures against DPA–a statistical analysis of their effectiveness. In: Topics in cryptology–CT-RSA 2004. Springer, pp 222–235
23. Edwards C (2015) Growing pains for deep learning. Commun ACM 58(7):14–16
24. Schneider T, Moradi A (2015) Leakage assessment methodology. In: Cryptographic hardware and embedded systems–CHES 2015. Springer, pp 495–513
25. Piret G, Quisquater J-J (2003) A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In: Cryptographic hardware and embedded systems, CHES 2003. Springer, pp 77–88
26. Robisson B, Manet P (2007) Differential behavioral analysis. In: Cryptographic hardware and embedded systems - CHES 2007, 9th international workshop, Vienna, Austria, September 10-13, 2007, Proceedings, pp 413–426
27. Blömer J, Seifert J-P (2003) Fault based cryptanalysis of the advanced encryption standard (AES). In: Wright RN (ed) Financial cryptography, volume 2742 of lecture notes in computer science. Springer, pp 162–181
28. Malkin T, Standaert FX, Yung M (2005) A comparative cost/security analysis of fault attack countermeasures. In: 2005 workshop on fault diagnosis and tolerance in cryptography (FDTC). IEEE, pp 109–123
29. Maistri P, Leveugle R (2008) Double-data-rate computation as a countermeasure against fault analysis. IEEE Trans Comput 57(11):1528–1539
30. Joye M, Manet P, Rigaud J-B (2007) Strengthening hardware AES implementations against fault attacks. IET Inf Secur 1(3):106–110
31. Karri R, Kuznetsov G, Goessel M (2003) Parity-based concurrent error detection of substitution-permutation network block ciphers. In: Cryptographic hardware and embedded systems-CHES 2003. Springer, pp 113–124
32. Patranabis S, Chakraborty A, Mukhopadhyay D (2015) Fault tolerant infective countermeasure for AES. In: Security, privacy, and applied cryptography engineering. Springer, pp 190–209
33. Bringer J, Carlet C, Chabanne H, Guilley S, Maghrebi H (2014) Orthogonal direct sum masking. In: Information security theory and practice. Securing the internet of things. Springer, pp 40–56
34. Danger J-L, Guilley S, Bhasin S, Nassar M, Sauvage L (2009) Overview of dual rail with precharge logic styles to thwart implementation-level attacks on hardware cryptoprocessors. SCS 00431261:1–7
35. He W, Breier J, Bhasin S, Chattopadhyay A (2016) Bypassing parity protected cryptography using laser fault injection in cyber-physical system. In: Proceedings of the 2nd ACM international workshop on cyber-physical system security. ACM, pp 15–21
36. Junod P, Vaudenay S (2005) Perfect diffusion primitives for block ciphers. In: Selected areas in cryptography. Springer, pp 84–99
37. Brinkmann M, Leander G (2008) On the classification of APN functions up to dimension five. Des Codes Crypt 49(1-3):273–288
38. Mukhopadhyay D, Chowdhury DR (2011) A parallel efficient architecture for large cryptographically robust n × k (k¿n/2) mappings. IEEE Trans Comput 60(3):375–385
39. Anderson R, Biham E, Lars K (1998) Serpent: a proposal for the advanced encryption standard. NIST AES Proposal 174:349–354
40. Guo J, Peyrin T, Poschmann A (2011) The photon family of lightweight hash functions. In: Advances in cryptology–CRYPTO 2011. Springer, pp 222–239
41. Guo J, Peyrin T, Poschmann A, Robshaw M (2011) The led block cipher. In: Cryptographic hardware and embedded systems–CHES 2011. Springer, pp 326–341
42. Augot D, Finiasz M (2014) Direct construction of recursive MDS diffusion layers using shortened BCH codes. In: Fast software encryption. Springer, pp 3–17
43. Sim SM, Khoo K, Oggier F, Peyrin T (2015) Lightweight MDS involution matrices. In: International workshop on fast software encryption. Springer, pp 471–493

44. Li Y, Wang M (2016) On the construction of lightweight circulant involutory MDS matrices. In: International conference on fast software encryption. Springer, pp 121–139
45. Liu M, Sim SM (2016) Lightweight MDS generalized circulant matrices. In: International conference on fast software encryption. Springer, pp 101–120
46. Sarkar S, Syed H (2016) Lightweight diffusion layer: importance of toeplitz matrices. IACR Transactions on Symmetric Cryptology 2016(1):95–113
47. Jean J, Peyrin T, Sim SM, Tourteaux J (2017) Optimizing implementations of lightweight building blocks. IACR Transactions on Symmetric Cryptology 2017(4):130–168
48. Avanzi R (2017) The QARMA block cipher family. almost mds matrices over rings with zero divisors, nearly symmetric even-mansour constructions with non-involutory central rounds, and search heuristics for low-latency S-Boxes. IACR Transactions on Symmetric Cryptology 2017(1):4–44
49. Chen C-L, Hsiao MY (1984) Error-correcting codes for semiconductor memory applications: a state-of-the-art review. IBM J Res Dev 28(2):124–134
50. Costello D, Lin S (2004) Error control coding. New Jersey
51. Rodríguez-Henríquez F, Saqib NA, Perez AD, Koc CK (2007) Cryptographic algorithms on reconfigurable hardware. Springer, Berlin
52. Mukhopadhyay D, Chakraborty RS (2014) Hardware security : design, threats, and safeguards. CRC Press, Boca Raton
53. Guilley S, Hoogvorst P, Pacalet R (2004) Differential power analysis model and some results. In: Quisquater JJ, Paradinas P, Deswarte Y, El Kalam AA (eds) Smart card research and advanced applications VI – CARDIS 2004. Kluwer Academic Publishers, pp 127–142
54. Prouff E (2005) DPA attacks and S-Boxes. In: Handschuh H, Gilbert H (eds) Fast software encryption – FSE 2005, volume 3557 of lecture notes in computer science. Springer, pp 424–442
55. Whitnall C, Oswald E (2011) A comprehensive evaluation of mutual information analysis using a fair evaluation framework. In: Rogaway P (ed) CRYPTO, volume 6841 of lecture notes in computer science. Springer, pp 316–334
56. Kavun EB, Yalçin T (2011) RAM-based ultra-lightweight FPGA implementation of PRESENT. In: 2011 international conference on reconfigurable computing and FPGAs, ReConFig 2011, Cancun, Mexico, November 30 - December 2, 2011, pp 280–285
57. Roy DB, Das P, Mukhopadhyay D (2015) ECC on your fingertips: a single instruction approach for lightweight ECC design in GF(p). In: Selected areas in cryptography - SAC 2015 - 22nd international conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers, pp 161–177
58. Borghoff J, Canteaut A, Güneysu T, Kavun EB, Knezevic M, Knudsen LR, Leander G, Nikov V, Paar C, Rechberger C, Rombouts P, Thomsen SS, Yalçin T (2012) PRINCE - a low-latency block cipher for pervasive computing applications - extended abstract. In: Wang X, Sako K (eds) Advances in cryptology - ASIACRYPT 2012 - 18th international conference on the theory and application of cryptology and information security, Beijing, China, December 2-6, 2012. Proceedings, volume 7658 of lecture notes in computer science. Springer, pp 208–225
59. Dobraunig C, Eichlseder M, Mangard S, Mendel F (2014) On the security of fresh re-keying to counteract side-channel and fault attacks. In: International conference on smart card research and advanced applications. Springer, pp 233–244
60. Chen C, Farmani M, Eisenbarth T (2016) A tale of two shares: why two-share threshold implementation seems worthwhile-and why it is not. IACR Cryptology ePrint Archive 2016:434
61. Poschmann A, Moradi A, Khoo K, Lim C-W, Wang H, Ling S (2011) Side-channel resistant crypto for less than 2,300 GE. J Cryptol 24(2):322–345
62. Moradi A, Wild A (2015) Assessment of hiding the higher-order leakages in hardware - what are the achievements versus overheads? In: Cryptographic hardware and embedded systems - CHESS 2015 - 17th international workshop, Saint-Malo, France, September 13-16, 2015, Proceedings, pp 453–474
63. Sasdrich P, Moradi A, Güneysu T (2015) Affine equivalence and its application to tightening threshold implementations. In: Selected areas in cryptography - SAC 2015 - 22nd international conference, Sackville, NB, Canada, August 12-14, 2015, revised selected papers, pp 263–276
64. De Cnudde T, Nikova S (2016) More efficient private circuits II through threshold implementations. In: 2016 workshop on fault diagnosis and tolerance in cryptography, FDTC 2016, Santa Barbara, CA, USA, August 16, 2016, pp 114–124
65. Cnudde TD, Nikova S (2017) Securing the present block cipher against combined side-channel analysis and fault attacks. IEEE Trans Very Large Scale Integr VLSI Syst 25:3291–3301
66. Schneider T, Moradi A, Güneysu T (2016) Parti - towards combined hardware countermeasures against side-channel and fault-injection attacks. In: Advances in cryptology - CRYPTO 2016 - 36th annual international cryptology conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II, pp 302–332
67. Guo J, Peyrin T, Poschmann A, Robshaw MJB (2011) The LED block cipher. In: Cryptographic hardware and embedded systems - CHES 2011 - 13th international workshop, Nara, Japan, September 28 - October 1, 2011, Proceedings, pp 326–341

Affiliations

Sikhar Patranabis[1] · Debapriya Basu Roy[1] · Anirban Chakraborty[1] · Naveen Nagar[2] · Astikey Singh[2] · Debdeep Mukhopadhyay[1] · Santosh Ghosh[3]

Debdeep Mukhopadhyay
debdeep@cse.iitkgp.ac.in

Santosh Ghosh
santosh.ghosh@intel.com

[1] Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, 721302, West Bengal, India

[2] Department of Electrical Engineering, Indian Institute of Technology Kharagpur, Kharagpur, 721302, West Bengal, India

[3] Intel Labs, Intel Corporation 2111 NE 25th Ave, Hillsboro, OR 97124, USA