Side-channel Resistant Soft Core Processor for Lightweight Block Ciphers

William Diehl, Abubakr Abdulgadir, Jens-Peter Kaps and Kris Gaj
Department of Electrical and Computer Engineering
George Mason University
Fairfax, U.S.A.
{wdiehl, aabdulga, jkaps, kgaj}@gmu.edu

***Abstract*— Lightweight cryptographic algorithms which provide moderate security at low cost, especially in very-light power-, energy, and resource-constrained processors, are an important topic of research in the context of the Internet of Things (IoT). Current cryptographic contests and standardization efforts seek to evaluate side-channel resistance of lightweight ciphers on multiple platforms, including resource-constrained 8-bit microprocessors. Using a custom-designed reconfigurable soft core processor on an FPGA, we implement four ciphers, SIMON, PRESENT, LED, and TWINE, and evaluate them for vulnerability to differential power analysis (DPA) using the t-test leakage detection methodology and an open-source test bench (FOBOS). We then adapt and modify techniques used in previous cipher hardware implementations to protect the soft core processor against 1st order DPA. Improved resistance to DPA is verified using the t-test and the FOBOS test bench. No modifications to cipher source code are required for the protected soft core, meaning that software programmers are insulated from the requirement to learn side-channel resistance techniques. A single DPA-resistant soft core instance, which can load and protect all four ciphers simultaneously (where target cipher is selected by the user at run-time), uses 392 slices in the Virtex-7 FPGA – an average of 98 slices per protected cipher.**

Keywords*— *Cryptography, side channel attack, field programmable gate array, reconfigurable, microcontroller, t-test

I. Introduction

THE range of potential internet devices has increased exponentially in recent years with the "Internet of Things" (IoT). As many devices in the IoT require some level of security of transactions, cryptographic applications are required to migrate to new families of devices. These new devices in the IoT are often heavily constrained by resources, such as power, energy, and space, and can include radio-frequency identification (RFID) tags, micro-authentication devices for automotive and aviation applications, home appliances, biometric devices, etc. Many such applications can be secured using lightweight cryptographic algorithms.

Lightweight cryptographic solutions can be hosted on a variety of platforms, including ASIC, FPGA, high-end CPUs, or lightweight microprocessors. In this research, we focus on the latter platform, namely lightweight 8-bit microprocessors.

Emphasis on 8-bit processors as an implementation platform is supported by current cryptographic contests and standards development projects that target improvements in lightweight cryptography. One example is the Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR), which calls for submissions and evaluations of candidate authenticated ciphers [1]. The CAESAR committee specified use cases for which candidates would be optimized and ultimately selected during final rounds [2]. One of these use cases is lightweight applications (resource constrained environments). The desired characteristics for authenticated ciphers conforming to this use case include performance on 8-bit CPUs (particularly for very short messages), and resistance to side-channel attacks [2].

In a second example, the National Institute of Standards and Technology's (NIST) Lightweight Cryptography Project is developing new recommendations using an open call for proposals to standardize algorithms. Algorithms will be evaluated on several characteristics, to include physical (area, memory, implementation type), performance (latency, throughput, power), and security (including side-channel resistance) [3].

In strong cryptography, the best-known cryptanalytic attack is no easier than a "brute-force" attack, i.e., the adversary is required to try an average of 50% of all possible keys until the correct key is identified. Such attacks are generally computationally infeasible for sufficient key sizes. However, physical cryptography, i.e., cryptographic processes hosted on any platform, is vulnerable to leakage of sensitive information through side-channel attack, such as differential power analysis (DPA), which can recover all or part of sensitive variables [4].

Accordingly, microprocessors, including lightweight processors targeted for lightweight cryptography, must be designed to minimize leakage of sensitive information through DPA. In this work, we support the CAESAR and NIST studies by instantiating four lightweight secret-key block ciphers (i.e., SIMON, PRESENT, LED, and TWINE) in a custom-designed very lightweight reconfigurable 8-bit processor. This "soft core" processor is implemented on the Spartan-3E FPGA, and tested for vulnerability using the t-test leakage detection methodology and the FOBOS power analysis architecture during the operation of the above ciphers.

TABLE I

BLOCK CIPHER VARIANTS IMPLEMENTED IN THIS RESEARCH

Cipher	Block Size	Key Size	Rnds	Type
SIMON 96/96	96	96	52	Feistel, ARX
PRESENT 64/80	64	80	31	SPN
LED 64/80	64	80	48	SPN
TWINE 64/80	64	80	36	SPN

After demonstrating that all four ciphers are vulnerable to DPA, we modify the processor to make it side-channel resistant, with the goals of 1) the lowest cost of protection (measured in increased area, reduced maximum frequency, and increased cycles-per-byte); and 2) insulating the programmer from any details of protection by requiring that identical cipher source and object codes run on both protected and non-protected soft core versions.

We verify improved resistance to DPA using t-tests and FOBOS during cipher operation. Finally, we benchmark unprotected and protected soft cores on the Virtex-7 FPGA and quantify the cost of protection in terms of throughput (Mbps), area (LUTs/slices), throughput-to-area (TP/A) ratio, software cycles-per-byte, power (mW), energy-per-bit (nJ/bit), and required random bits-per-cycle.

II. BACKGROUND AND PREVIOUS WORK

A. Block ciphers implemented in this research

The major characteristics of the block cipher variants implemented in this research are shown in Table I. The internal structure of a regular cipher round for all investigated ciphers is shown in Figs. 1 and 2. The reader is referred to [5 – 8] for the detailed specifications of all implemented ciphers, including any additional information on pre-processing (computations performed before the first round), post-processing (computations executed after the last round), special rounds (e.g., a reduced-functionality last round), and key scheduling (i.e., calculation of round keys).

These ciphers are used as cryptographic primitives for authenticated ciphers being evaluated in the CAESAR third round competition, including CLOC-TWINE, SIMON-JAMBU, SILC-PRESENT, and SILC-LED [9, 10].

B. Previous implementations of ciphers in this research

Ciphers in this research have been previously implemented in microcontrollers and reconfigurable processors, including the 8-bit AVR ATmega128, and the 16-bit TI MSP-430, in the FELICS and BLOC studies, respectively [11, 12]. In [13], the authors compare six block ciphers (including SIMON, PRESENT, LED, and TWINE) in terms of area, performance, power, and energy-per-bit, using both hardware (i.e., register-transfer level) and software (i.e., custom-designed soft core processor) implementations running in the same FPGA.

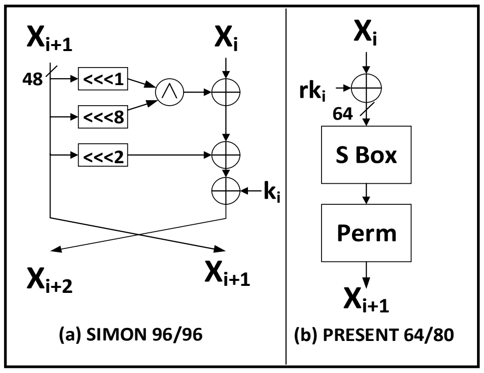

Fig. 1 (a) SIMON 96/96 (b) PRESENT 64/80

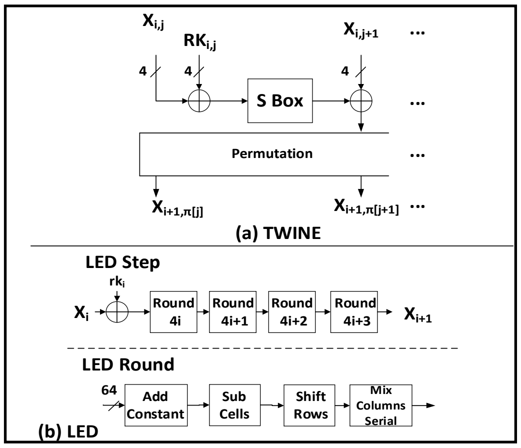

Fig. 2 (a) TWINE 64/80 (b) LED 64/80

Custom-designed processors have also been used as testbeds for side-channel protection of cryptographic algorithms. A relevant example is [14], in which the authors protect the datapath of a custom-processor designed to accommodate Addition, Rotation, and XOR (ARX) ciphers, using a 3-share threshold implementation-based ALU, and verify resistance to 1st order DPA using the t-test leakage detection methodology.

C. Leakage Detection Methodology: Welch's t-test

Differential Power Analysis (DPA), pioneered in [4] and substantially improved in subsequent decades [15], is used to analyze differences between observed power measurements, and hypothetical power (based on presumed contents of a sensitive variable) according to a power model. However, coming up with the power model is difficult, time consuming, and requires expert knowledge of the underlying architecture and technology [16].

An expedited leakage assessment methodology proposed in [16], and further described in [17], uses the Welch's t-test to determine whether two distributions are different from one another. Some of the advantages in using the t-test for an assessment of leakage are 1) The t-test finds leakage of information without mounting an attack, 2) The test does not rely on knowledge of the underlying architecture, and 3) The t-test can quickly reveal when information leaks and when a countermeasure has failed. However, it is not a complete substitution for DPA. For example, 1) No information is provided about the difficulty of mounting an attack, and 2) There is no recovery of a key, message, sensitive intermediate values, or the correct power model.

In the Welch's t-test, a confidence factor t is calculated as $t = (\mu_0 - \mu_1)/\sqrt{{s_0}^2/n_0 + {s_1}^2/n_1}$, where μ_0 and μ_1 are means of distributions Q_0 and Q_1, s_0 and s_1 are standard deviations, and n_0 and n_1 are the cardinality of the distributions, or the number of samples. An additional calculation of degrees of freedom (v) is not required if $n_0 = n_1$, and the number of samples is sufficiently large (e.g., greater than 1000).

Given a normally distributed probability density function (pdf) $f(t, v)$, a probability p is calculated as $p = 2\int_{|t|}^{\infty} f(t, v)dt$. To use the t-test, we start with two distributions. We assume a null hypothesis – namely, that "samples are drawn from the same population," and that "samples are not distinguishable." We can estimate the result of the t-test as $p = 2F(-|t|, v)$, where F is the hypergeometric function corresponding to the pdf. Finally, we designate a "threshold." For example, we assume that $|t| > 4.5$ is defined to reject the null hypothesis. This assumption is based on the fact that $p = 2F(-4.5, n > 1000) < -0.00001$, which means that we can reject the null hypothesis with 99.999% confidence. In summary, we search for a result where $|t| > 4.5$. If this occurs during analysis of the two distributions, we reject the null hypothesis that "the samples are from the same distribution" and reason that the device is leaking information.

If our goal is to plausibly show that a device is leaking information (without a specific need to recover a sensitive variable or demonstrate the difficulty of an attack), we can use the so-called "non-specific t-test." In the non-specific t-test, we preselect some sensitive "fixed" data D (e.g., message). Then we randomly interleave the feeding of D, or random data, to the algorithm. We call this characterization a "fixed versus random" test [17].

D. Threshold Implementations

Masking is a well-known countermeasure against power analysis side channel attack. However, masking can fail when implemented in CMOS technology, since the power change that occurs in a CMOS gate during a transition due to a glitch is relatively large compared to normal operation of a device. Measuring the toggle rate of CMOS glitches has been used to successfully attack a masked version of AES [18].

One method of providing security in the presence of glitches is called threshold implementations (TI) [19]. In order to be provably secure against power analysis in the presence of glitches, a threshold implementation must have the following three properties:

Property 1 - Non-completeness. Every function is independent of at least one share of each of the input variables. Formally, if $z = N(x, y)$ and x and y are in n shares, then

$$z_1 = f_1(x_2, x_3, \dots, x_n, y_2, y_3, \dots, y_n), \quad (1)$$

$$z_2 = f_2(x_1, x_3, \dots, x_n, y_1, y_3, \dots, y_n), \quad (2)$$

$$z_n = f_n(x_1, x_2, \dots, x_{n-1}, y_1, y_2, \dots, y_{n-1}). \quad (3)$$

In other words, If z_i does not depend on x_i and y_i, it cannot leak information about x_i or y_i.

Property 2 – Correctness. The sum of the output shares gives the desired output. Formally, $z = \bigoplus_{i=1}^{n} z_i = N(x, y)$.

Property 3 – Uniformity. A realization of $z = N(x, y)$ is uniform if for all distributions of the inputs x,y,…, the sharing preserves the output distribution. As a corollary, if the function prior to sharing is a permutation, the shared function should also be a permutation.

To share a function of degree d, $d + 1$ shares are required. For example, a quadratic non-linear function (e.g. $z = xy$) can be shared using 3 shares. However, producing TI which achieve Property 3 is challenging, and often increases the cost of threshold implementations [20]. Uniformity can be achieved by supplying fresh masks inside pipelined stages, which is called "resharing" or "remasking." However, there is a cost in terms of increased pipelining stages, increased number of clock cycles, increased hardware, and increased requirement to provide sources of fresh randomness.

E. Our Contribution

Developing side-channel resistant microprocessors, and conducting analysis of vulnerability to side-channel attack of cryptographic algorithms, are both difficult tasks. In this research, we apply a combination of threshold implementation techniques that have been successfully demonstrated in RTL implementations of ciphers toward protection of a reconfigurable soft core microprocessor, capable of protecting multiple ciphers at once. We expand on [14] by simultaneous protection of several types of block ciphers (i.e., not only ARX ciphers), and reduce required FPGA resources through a hybrid 2- / 3- share TI-protected data bus, and other innovations.

Additionally, we demonstrate a use-case of the t-test leakage detection methodology to streamline the analysis, engineering, and verification of countermeasures in multiple ciphers – a historically long and difficult task using traditional methods of correlation power analysis (CPA).

The resulting product, the side-channel resistant soft core processor, provides a low-cost secure cryptographic platform (especially when run time flexibility is desired), and provides a template for the application of heterogeneous countermeasures to protect complex microprocessors against side-channel attack.

Finally, our model of the use of identical source and object codes on both unprotected and protected soft cores leads to greater security, in that the programmer is not required to learn complex leakage avoidance techniques.

III. Methodology

A. Custom-designed reconfigurable soft core microprocessor

Our soft core processor is a Reduced Instruction Set Computer (RISC) with 30 native instructions, draws 8-bit program words from program RAM, and uses an 8-bit data bus to address a separate data RAM in a Harvard Architecture. It is a pure load-store architecture, where no immediate writes are permitted to memory, and there are no ALU immediate addressing modes. Most instructions execute in one clock cycle, however, some ALU instructions execute in two cycles.

There are two enhancements to the processor designed to support cryptographic secret-key ciphers. The first provides up to four user-defined transformations (TRF). TRFs execute in one clock cycle and perform a transformation on a single (up to) 8-bit operand. TRFs are designed to facilitate cryptographic transformations such as S-Boxes or round constants, and are implemented in our default design as look-up table ROMs.

The second enhancement is "instruction set extensions" (ISE), which allow for up to eight user-defined ALU instructions. The ISEs execute in two clock cycles, use the ALU data bus, and perform operations on two operands.

The user can instantiate up to 4 Kbytes of program RAM, 64 Kbytes of data RAM (limited to 256 bytes in this research), and 1 Kbyte of table ROM (i.e., four 256-byte look up tables). Memory is implemented as Distributed RAM, and is instantiated at synthesis time. The amount of memory instantiated, based on the required number of bytes (req_bytes),

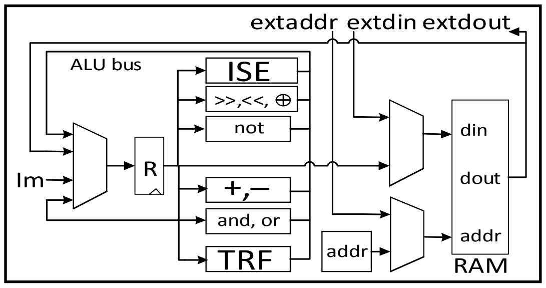

Fig. 3 Soft Core Datapath. R denotes "registers;" >> and << denote shifts or rotations; "Im" is immediate operand. All bus widths are 8 bits.

is computed as $2^{\lceil \log_2 req\_bytes \rceil}$. The use of Distributed RAM (which is instantiated using LUTs) is primarily a benchmarking technique, since a measure of effectiveness such as "throughput-to-area ratio" would be less meaningful if Block RAM (BRAM) resources were used.

This soft core processor has a simple but flexible interface consisting of 71 ports, allowing a higher-level entity the ability to load programs and data, specify start and stop points, control execution, and read data memory during and after program execution. A simplified depiction of the ALU and data RAM portion of the soft core datapath is shown in Fig. 3.

B. Development, Verification, Benchmarking, and Implementation

Each software code is developed using custom native assembly language. A Python assembler "SoftAsm" is used to produce object code which is directly portable to a runtime simulator "SoftSim," and to a VHDL module to be loaded into a synthesis tool, such as Vivado Design Suite.

The object code, RAM, and ROM tables are then instantiated in a VHDL loader in Vivado Design Suite. Correct results are verified using a simulator (such as iSim or Vivado Simulator), and synthesis and implementation results are generated using Xilinx ISE, Vivado or other FPGA implementation tool set.

The source files for the soft core processor, assembler and simulator, and reference manuals, are available at [21].

C. Unprotected Soft Core

"SoftAsm" is capable of combining multiple source files into one executable object file, including resolution of namespace and label conflicts. We have utilized this feature to compile a single object code for all four ciphers in this research, in order to allow simultaneous loading of all ciphers in a single soft core instance. Together the ciphers total 999 program and 256 data bytes (each cipher is allotted a 64-byte partition). This allows all ciphers to be instantiated in 1K of Program RAM.

Although the four ciphers are loaded simultaneously, only one can execute at a time. The execution sequence is controlled by a higher layer of protocol which provides the core with a start address, stop address, and a start signal.

We utilize all four transformations (TRFs) as follows: 1) TRF0 is used for the PRESENT and LED S-Box (recalling that PRESENT and LED use the same S-Box); 2) TRF1 is used for the TWINE S-Box; 3) TRF2 is used for LED round constants; and 4) TRF3 is used for TWINE round constants.

We use instruction set extensions (ISE) for both PRESENT and LED. In the case of PRESENT, incorporation of two specific extensions, a hardware permutator and 61-bit rotator, reduces total clock cycles from 77,225 to 20,030 – a 74% reduction. The number of look-up tables (LUTs) required to accommodate this soft core actually decreases by about 2%. This is due to a reduction in program and data size which reduces the required instantiation of program and data RAMs.

ISEs also facilitate the processing of LED. The results of previous microcontroller benchmarking studies show that many designers have trouble with LED [11, 12]. We implement a column vector multiplier which reduces the cycle count from 46,817 to 30,015 – a 36% decrease. There is a corresponding increase of 7% in the number of required LUTs.

D. Flexible Open-source workBench fOr Side-channel analysis (FOBOS)

FOBOS is a free and open tool which provides a single "acquisition to analysis" solution to measure resistance to power analysis side-channel attack (SCA) and evaluation of the effectiveness of countermeasures [22]. FOBOS leverages low-cost hardware, such as the Diligent Nexys 2 and Xilinx Spartan 3E FPGA Starter Board.

A complete description of FOBOS capabilities is available at [23]. We start with the baseline FOBOS software suite available at [23], and modify the analysis tool set to perform the non-specific t-tests as described above.

E. Protected Soft Core

Our implementations are constant-time, with no branches dependent on sensitive data, and therefore are resistant to timing and simple power analysis (SPA) attacks.

To protect against DPA, we leverage [19, 20] to implement a hybrid 2- /3- share threshold implementation of the ALU and data RAM portion of the datapath (no change is required to the program datapath). We duplicate the register and data RAM banks, and label them "A-bus" and "B-bus." We also expand the *extdin* port to *extdina* and *extdinb*, and *extdout* port to *extdouta* and *extdoutb*. Data that enters the soft core should be logically pre-separated into two shares, using 1st order Boolean masking. There is only one register selector and address computer for registers and memory, respectively. Registers and data memory on both buses are always read and written in tandem in this architecture. The protected architecture is shown in Fig. 4.

Within the protected datapath, three types of operations can take place: 1) Linear operations, such as shifts, rotations, `xor`, `not`, etc. These are performed by duplicating hardware logic to allow each bus to access its own logically-separated devices. The result remains on the originating bus; 2) Non-linear operations, such as `and`, `or`, certain TRFs (such as S-Boxes). These are performed by specially-constructed 3-share TI-protected components, as described below. These devices require inputs from the A- and B- buses, and return masked results to both buses; and 3) "Quarantined" operations. At times, instructions require data from both buses to make decisions, such as determining the proper memory address for reads or writes, computing shift distances, or processing status flags (such as N or Z flags). In this implementation, the processor does not know which data is sensitive or non-sensitive. However, it can be designed to "quarantine" data based on knowledge of the instruction and its intended use. The

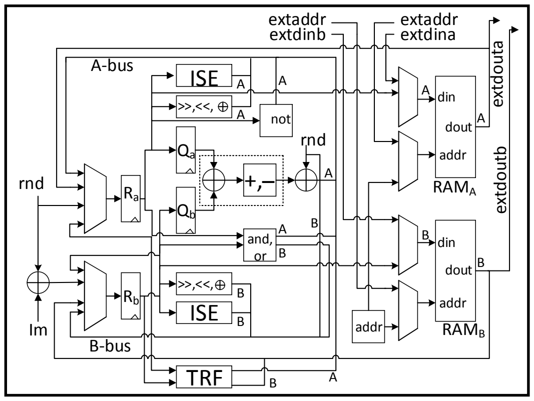

Fig. 4 Soft Core Protected Datapath. $R_{a,b}$ denote "registers," $Q_{a,b}$ are "quarantine" registers, >> and << denote shifts or rotations, "Im" is immediate operand, "rnd" denotes random bits. All bus widths are 8 bits.

quarantine method isolates key operands inside registers, stalls the processor for one cycle, and then allows computations to occur on both operands using non-protected circuits.

In the case of these four ciphers, there are no additions on sensitive data (examples of ciphers which use addition for non-linearity are SPECK and RC6). Therefore, we have not implemented a TI-protected adder in the protected soft core. Our rationale is that such an adder is costly. For example, an 8-bit 3-share TI adder based on [24, 25] costs at least 120 LUTs in the Virtex-7. Accordingly, we implement addition functions as quarantined functions, since `add`, `sub`, `inc`, and `dec` are used for program flow control.

We do require a 3-share TI-protected `and` for use in SIMON, which is in the ARX family of ciphers. Strictly speaking, we do not require a TI-protected `or`. Most of the ciphers in this research use the `or` command, however, its intended effect is concatenation – not a non-linear transformation. However, variable concatenations (i.e., those not bounded at run time) can be costly to implement in hardware. Additionally, a 3-share TI-protected `or` is able to share most components with its corresponding protected `and`. Therefore, we build a combined 3-share TI-protected `and-or` device.

The output of the combined `and-or` is not a permutation of input bits – random refresh bits must be provided on each invocation to ensure the TI uniformity property.

F. Protected Transformations

We modify the PRESENT and LED (TRF0), and TWINE (TRF1) S-Boxes, respectively, with 3-share TI-protected S-Boxes.

For the PRESENT and LED S-Boxes, we implement a two-cycle 3-share TI-solution adapted from [26, 27]. This 4-bit S-Box includes a permutation *G* of quadratic order, which enables a 3-share TI. The function *G*, along with affine transformations, is cascaded to form a composite function to represent the PRESENT S-Box, which is of cubic order. This requires four bits of randomness per invocation to increase from two to three shares, but does not require additional randomness in the permutation. The PRESENT and LED protected S-Box (TRF0) is shown in Fig. 5.

The TWINE 4-bit S-Box is also of cubic order. To achieve a three-share TI we employ a strategy previously used for AES in [28]. According to Fermat's Little Theorem, $a^p \equiv a \bmod p$,

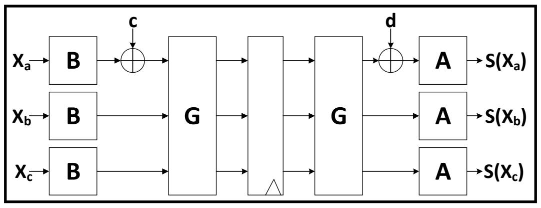

Fig. 5 3-share TI-protected S-Box used in TRF0 for PRESENT and LED. A and B are matrix multiplications; c and d are constants; G is a quadratic composite function. All bus widths are 4 bits.

and $a^{p-2} \equiv a^{-1} \bmod p$. In this case, we can compute $x^{14} \equiv x^{-1}$ in GF(2^4). This decomposes into two cascaded multipliers of quadratic order, which enables our three-share TI.

The cascaded multipliers on GF(2^4) are not permutations – they do not satisfy the TI uniformity property. Refreshed masking is required at each of the two levels to ensure this property. MATLAB Monte Carlo simulations confirm that uniformity is achieved with one random bit per 4-bit multiplier, for a total of two bits per S-Box, plus four bits for increasing from two to three shares. The three-share inverter used in TRF1 is shown in Fig. 6.

Our two-stage cascaded 3-share TI transformations in TRF0 and TRF1 are not required to be registered. However, since we are forced to pay a one-cycle penalty in TRF2 and TRF3 to enforce a quarantine, we insert registers between stages in TRF0 and TRF1. This has the effect of further increasing security by minimizing glitch propagation chains, and keeps the soft core's critical path as short as possible.

G. Protected Instruction Set Extensions

The instruction set extensions (ISE) for PRESENT and LED are easily protected in a 2-share TI scheme, since both are linear transformations. Each corresponding hardware device is duplicated to appear on both the A- and B-bus. However, all PRESENT and LED ISEs use one operand as a pointer to select the target register. To compute this target without information leakage, these ISE must all be quarantined, which causes a one-cycle penalty.

H. Optimization of FPGA Results

Achieving optimum throughput-to-area (TP/A) ratios in Vivado is not trivial, as synthesis and implementation strategies attempt innovative placement and routing in order to meet a user-defined clock constraint. Maximum frequency is often estimated by iteratively varying the clock period until an acceptable worst negative slack (WNS) target (e.g., 0.1 ns) is achieved. However, this binary search procedure is time-consuming, and there is no guarantee that the user will achieve optimal throughput, area, or throughput-to-area ratio.

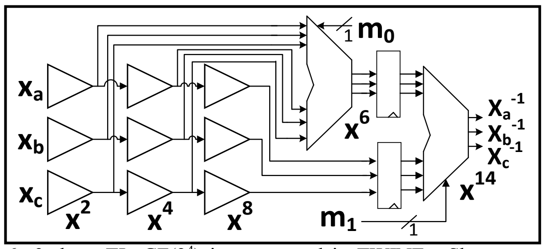

Fig. 6 3-share TI GF(2^4) inverter used in TWINE. Shares x_a, x_b, x_c are reshared from A- and B-bus. x^2, x^4, x^8 (produced by squares), and x^6 (where $x^6 = x^2 \cdot x^4$) are intermediate products; x^{14} (where $x^{14} = x^6 \cdot x^8$) is final product; All bus widths are 4 bits, except for random bits m_0 and m_1, which are single bits.

This research uses an optimization tool called Minerva to maximize TP/A ratios [29]. Minerva is a Python-based batch processor that spawns multiple Vivado sessions. It iteratively experiments with all 25 Vivado default implementation strategies until it finds the best TP/A ratio. Minerva is used for all implementations in this research, and achieves an average of 15% improvement in TP/A ratios.

IV. Results

The t-test leakage methodology, using 2000 high fidelity traces (i.e., 20,000 samples per trace), is performed on the four ciphers using the FOBOS architecture (using the Spartan 3E FPGA). The results, as shown in Fig. 7, show that all four unprotected ciphers fail the non-specific "fixed-versus-random" t-test, and are vulnerable to 1st order DPA.

The reconfigurable processor is then protected using the methods described in Section III. The same non-specific t-tests as described above are performed on the protected soft core. The t-tests all pass, meaning that $|t| < 4.5$ for all observed samples. The t-tests of the protected versions are shown in Fig. 8.

The protected processor increases the number of cycles required for certain instructions. Table II shows the program memory requirements, total clock cycles, and cycles-per-byte for each cipher for the unprotected and protected soft cores.

The unprotected and protected reconfigurable processors are implemented in the Virtex-7 FPGA, and optimized for throughput-to-area (TP/A) ratio by Minerva. Table III shows the implementation statistics for the unprotected and protected soft core processors, as implemented in this research.

The actual power and energy-per-bit of unprotected and protected ciphers are measured during operation on the Spartan 3E using a combination of the FOBOS architecture, measurement of voltage across a shunt resistor, and amplification of the voltage. The results, computed at a fixed external clock frequency of 5 MHz, are shown in Table IV.

V. Analysis

A. Results in this work

The protected soft core passes all non-specific t-tests, which shows that the design is resistant to 1st order DPA. Although the unaltered software executes on both platforms, there are penalties in terms of clock cycles when using the protected design. Analysis from Table II shows an average increase of 25% in the number of required cycles and cycles-per-byte.

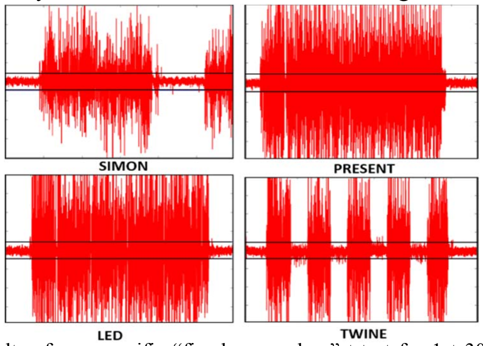

Fig. 7 Results of non-specific "fixed-vs.-random" t-test for 1st 3000 clock cycles of the four indicated ciphers on unprotected soft core, with time-domain on x-axis, and t-values on y-axis; horizontal lines indicate |t|<4.5.

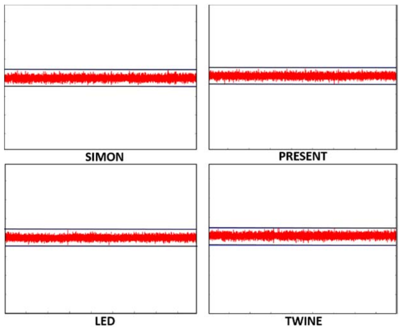

Fig. 8 Results of non-specific "fixed-vs.-random" t-test for 1st 3000 clock cycles of the four indicated ciphers on protected soft core, with time-domain on x-axis, and t-values on y-axis; horizontal lines indicate |t|<4.5.

TABLE II
SOFTWARE BENCHMARKING RESULTS FOR CIPHERS IN THIS RESEARCH ON THE UNPROTECTED AND PROTECTED SOFT CORES

Cipher		Unprotected		Protected		
	Prog Bytes	Cycl/ Block	Cycl/ Byte	Cycl/ Block	Cycl/ Byte	Ratio
SIMON	274	58234	4853	78482	6540	1.35
PRESENT	321	77225	9653	-	-	-
w/Perm	191	26509	3314	-	-	-
w/Perm & Rot	157	20030	2504	23639	2955	1.18
LED	463	46817	5852	-	-	-
w/Col Mul	314	30015	3752	40690	5086	1.36
TWINE	253	19892	2487	22353	2794	1.12

TABLE III
RESULTS OF IMPLEMENTATION OF UNPROTECTED AND PROTECTED SOFT CORES ON VIRTEX-7 FPGA

Implementation	Area	Area	Freq	TP	TP/A
	LUTs	Slices	MHz	Mbps	Mbps/LUT
Unprotected (UnPr)	613	185	253	0.808	0.001319
Protected (Pr)	1314	392	234	0.634	0.000482
Ratio (Pr/UnPr)	2.14	2.12	0.92	0.78	0.37

The protected soft core uses 2.1 times as many LUTs and slices. There is also a reduction in maximum frequency, primarily due to increased routing delay associated with the more complex design. Using the PRESENT cipher as an example, there is a 22% reduction in throughput, and a 63% reduction in throughput-to-area (TP/A) ratio in the protected soft core compared to the unprotected soft core.

In the protected soft core, there is a 21% increase in average power (which includes static and dynamic power), and 54% increase in required energy-per-bit, in comparison with the unprotected soft core.

B. Comparison with previous results

Comparison of these cipher implementations with dedicated implementations on fixed microcontrollers is difficult, given that results on fixed processors do not take into account the increased flexibility of the reconfigurable processor. A comparison of our software results with those in Table V shows that our 8-bit soft core processor generally underperforms results on the AVR ATmega128, but usually outperforms results on the TI MSP-430.

While microprocessor cryptographic implementations cannot compete with dedicated RTL implementations in FPGA

TABLE IV

POWER AND ENERGY-PER-BIT FOR UNPROTECTED AND PROTECTED SOFT CORES AS MEASURED ON SPARTAN 3E FPGA @ 5MHZ

Cipher	Unprotected (UnPr)		Protected (Pr)		Ratio	
	Power	Energy	Power	Energy	(Pr/UnPr)	
	mW	nJ/bit	mW	nJ/bit	Power	Energy
SIMON	14.9	1808	18.2	2976	1.22	1.65
PRESENT	14.6	914	18.1	1337	1.24	1.46
LED	15.7	1473	18.4	2340	1.18	1.59
TWINE	15.3	951	18.0	1257	1.16	1.32
Mean	15.1	1286	18.2	1977	1.21	1.54

TABLE V

COMPARISON OF CYCLES PER BLOCK ON UNPROTECTED SOFTCORE AGAINST OTHER MICROCONTROLLER BENCHMARKING EFFORTS

Cipher	TW	FELICS	FELICS/ TW	BLOC	BLOC/ TW
SIMON	58,234	4,208	0.072	50,328	0.865
PRESENT	20,030	10,017	0.500	222,066	11.087
LED	30,015	67,414	2.245	467,558	15.578
TWINE	19,892	13,685	0.687	36,290	1.824

TW is "this work." FELICS data from [11], Scenario 0 AVR ATmega128 detailed results. Implementations are ASM except LED and TWINE (ASM results not available). SIMON results are for the 64/96 version (96/96 results not available). BLOC data from [12]. All results are for C implementations on TI MSP-430. The LED version is LED64.

or ASIC, there is value in the use of a reconfigurable soft core processor, particularly if 1) multiple changes are required to controllers during rapid prototyping, or 2) the choice of application must be dynamically reselected during runtime. In these cases, the user is provided with an FPGA-hosted core using an average of 46 slices per application, assuming sharing of soft core resources. This is comparable to or smaller than dedicated lightweight RTL results reported for SIMON and PRESENT, at 36 and 117 slices, respectively [30, 31].

While there are few reported 3-share TI implementations of these ciphers in FPGA, one example is a 96-slice TI-protected version of SIMON [32]. Given the simplicity of TI-protection in SIMON, it is likely that our average of 98 slices per 1st order DPA-resistant cipher is among the best available results.

C. *Failed t-tests*

The visual presentations of successful t-tests as shown in Fig. 8 are obligatory for research papers, but are admittedly uninteresting to the reader. The more interesting case is when a t-test "almost passes," or "barely fails."

Determining the cause of leakage in a protected reconfigurable processor it not trivial – it is much more difficult than finding leakage in a dedicated RTL implementation, since any data on any bus can potentially cause a leak, whether or not the data, or culpable hardware feature, is even intended for the cipher in which the leak is observed.

An example of the analysis process used to find leakage is shown in Fig. 9. In this approach, we use a "time-domain" strategy, where we try to correlate cycle execution based on known cycles after the FOBOS trigger signal, discrete points of leakage as identified in the t-test, and analysis of program execution, as produced by SoftSim.py and rendered in MATLAB. We determined that the leakage in this case was caused by failure to satisfy the uniformity property in the combined 3-share TI-protected `and-or` device, and rectified the problem by applying additional fresh masking during the calculation.

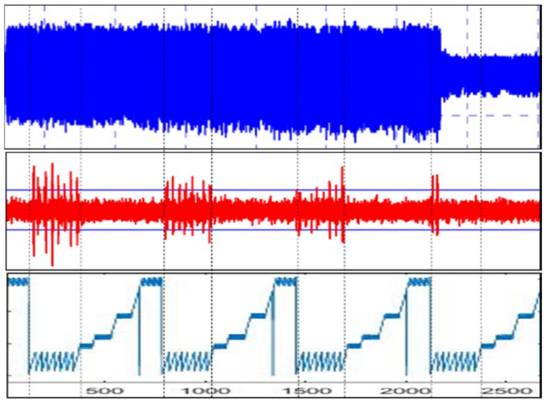

Fig. 9 Top window is time-domain of a FOBOS power trace. Center window is the t-test results, aligned with FOBOS power trace, with $|t|<4.5$ shown by horizontal lines. Bottom window is software program sequence. Dotted lines indicate time-aligned events. Time-domain is shown on x-axis in all cases.

D. *On required randomness*

Our protected soft core generates weak pseudorandom bits for remasking and resharing, using a combination of an LFSR, and rotations of pre-supplied random bits. The number of random bits required for each cipher in the protected soft core, amortized over all clock cycles, is shown in Table VI. An average of 2.2 random bits/cycle is required.

To achieve adequate security in practice, however, required randomness should be produced by true random number generators (TRNG). Ideally, one would seek to add a TRNG to the same FPGA. Although two random bits/cycle does not sound extreme, the requirement becomes 468 Mbps for a processor operating at 234 MHz. According to study in [33], the highest-throughput TRNG in the study, a self-timed ring (STR) TRNG, implemented on the Spartan 6 FPGA, is capable of only 154 Mbps. This TRNG would add about 350 LUTs to our protected design, resulting in a 27% growth in area. Therefore, the ability to source sufficient randomness, especially on-chip, is a critical design parameter that will influence the development of side-channel-resistant processors.

VI. CONCLUSION

In this research, we implemented four lightweight block ciphers, SIMON, PRESENT, LED, and TWINE, in a custom-designed "soft core" reconfigurable 8-bit processor. We were able to simultaneously load all four ciphers in one unprotected soft core instance using 185 slices, which average about 46 slices per cipher when implemented on the Virtex-7 FPGA. Using the t-test leakage detection methodology and the FOBOS power analysis test bench, we found that all four ciphers are vulnerable to differential power analysis (DPA) on an unprotected soft core.

We then leveraged technologies applied to register-transfer-level (RTL) implementations of DPA-resistant ciphers to protect our soft core using a hybrid 2/ - 3-share threshold

TABLE VI

NUMBER OF REQUIRED RANDOM BITS PER CYCLE IN PROTECTED SOFT CORE

Cipher	SIMON	PRESENT	LED	TWINE	Mean
Random bits/cycle	2.5	1.9	2.0	2.5	2.2

implementation (TI). Retesting of the four ciphers on the protected soft core, using the t-test and FOBOS architecture, confirms improved resistance to 1st order DPA. No changes in cipher source or object codes are required for the protected core – a feature which improves secure software design by not relying on programmer skill in leakage avoidance techniques.

Comparing the protected versus the unprotected soft cores, there is a 25% increase in required cycles/byte, an approximate two-fold increase in both LUTs and slices, a 22% reduction in throughput, and 63% reduction in throughput-to-area (TP/A) ratio, using PRESENT as a test case. The protected core uses an average of 21% more power and 54% more energy-per-bit for block cipher encryption than the unprotected core. The protected soft core uses 2.2 bits/cycle of randomness – a modest amount which nevertheless taxes state-of-the-art FPGA true random number generation (TRNG) capabilities.

Finally, the protected soft core achieves protection against 1st order DPA for the subject ciphers at 392 slices (an average cost of 98 slices per cipher) – a respectable result even among the best reported FPGA implementations of protected ciphers.

VII. Areas for Further Research

Future work could include higher-order DPA protection of these ciphers, and other lightweight ciphers (such as PRINCE, HIGHT, Piccolo, FANTOMAS, etc.), improved performance by incorporating advanced microprocessor techniques such as pipelining and superscalar architecture, incorporation of TRNG on the same FPGA, and comparison of results against DPA-protected versions of other very lightweight reconfigurable processors, such as the Xilinx PicoBlaze.

References

[1] "CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness." Internet: http://competitions.cr.yp.to/caesar.html.

[2] D. Bernstein, 2016, Jul. 16, Google Groups, "Cryptographic Competitions," https://groups.google.com/forum/#!forum/crypto-competitions

[3] K. McKay, L. Bassham, M. Turan and N. Mouha, "Report on Lightweight Cryptography (NISTIR 8114)," National Institute of Standards and Technology (NIST), 2017, Internet: http://nvlpubs.nist.gov/nistpubs/ir/2017/NIST.IR.8114.

[4] P. C. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *CRYPTO '99 - 19th International Conference on Cryptology*, Aug. 15-19, 1999, Santa Barbara, CA.

[5] R. Beaulieu, S. Treatman-Clark, D. Shors, B. Weeks, J. Smith and L. Wingers, "The SIMON and SPECK lightweight block ciphers," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, San Francisco, CA, 2015, pp. 1-6.

[6] A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, C. Vikkelsoe, P. Paillier and I. Verbauwhede, "PRESENT: An Ultra-Lightweight Block Cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2007: 9th International Workshop*, Vienna, Austria, September 10-13, 2007, pp. 450-466.

[7] J. Guo, T. Peyrin, A. Poschmann and M. Robshaw, "The LED Block Cipher,"in *Cryptographic Hardware and Embedded Systems (CHES 2011): 13th International Workshop*, Nara, Japan, Sep. 28 – Oct. 1, 2011, pp. 326-341.

[8] T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi, "TWINE: A Lightweight Block Cipher for Multiple Platforms," *SAC*, vol. 7707, 2012, pp. 339–354.

[9] H. Wu and T. Huang, "The JAMBU Lightweight Authenticated Encryption Mode v2.1," Sep. 15, 2016, https://competitions.cr.yp.to/round3/jambuv21.pdf

[10] T. Iwata, K. Minematsu, J. Guo, S. Morioka and E. Kobayashi, "CLOC & SILC v3," Sep. 15. 2016, https://competitions.cr.yp.to/round3/clocsilcv3.pdf

[11] D. Dinu, A. Biryukov, J. Großschädl, D. Khovratovich, Y. Le Corre and L. Perrin, "Fair Evaluation of Lightweight Cryptographic Systems (FELICS),"Lightweight Cryptographic Workshop, 2015, Internet: http://csrc.nist.gov/groups/ST/lwc-workshop2015/papers/session7-dinu-paper.pdf

[12] M. Cazorla, K. Marquet and M. Minier. "Survey and benchmark of lightweight block ciphers for wireless sensor networks," in *the 10th International Conference on Security and Cryptography* (SECRYPT 2013), Reykjavík, Iceland, Jul. 29-31, 2013, pp. 543-548.

[13] W. Diehl, F. Farahmand, P. Yalla, J. P. Kaps and K. Gaj, "Comparison of hardware and software implementations of selected lightweight block ciphers," *2017 27th International Conference on Field Programmable Logic and Applications* (FPL), Ghent, Belgium, 2017, pp. 1-4.

[14] F. Bache, T. Schneider, A. Moradi and T. Güneysu, "SPARX — A side-channel protected processor for ARX-based cryptography," *Design, Automation & Test in Europe Conference & Exhibition* (DATE), Lausanne, 2017, pp. 990-995.

[15] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to Differential Power Analysis," *Journal of Cryptographic Engineering*, Apr. 2011, vol. 1, pp. 5-27.

[16] G. Goodwill, B. Jun, J. Jaffe and P. Rohatgi, "A testing methodology for side channel resistance validation," NIST Non-invasive Attack Testing Workshop, 2011.

[17] T. Schneider and A. Moradi, "Leakage Assessment Methodology", *Journal of Cryptographic Engineering*, Jun. 1, 2016, vol. 6, pp. 85-89.

[18] S. Mangard, N. Pramstaller and E. Oswald, "Successfully attacking masked AES hardware implementations" *CHES 2005*. LNCS, vol. 3659, pp. 157–171.

[19] S. Nikova, C. Rechberger and V. Rijmen, "Threshold Implementations Against Side-Channel Attacks and Glitches," *Information and Communications Security*, LNCS, vol. 4307, 2006, pp. 529-545.

[20] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov and V. Rijmen, "A More Efficient AES Threshold Implementation," in *7th International Conference on Cryptology in Africa* (AFRICACRYPT 2014), Marrakesh, Morocco, May 28-30, 2014, pp. 267-284.

[21] W. Diehl, "Custom Very-lightweight 8-bit Soft Core Processor," Oct. 20, 2017, Internet: https://cryptography.gmu.edu/athena/softcore

[22] R. Velegalati and J.P. Kaps, "Towards a Flexible Opensource Board for Side-channel analysis (FOBOS)," *Cryptographic architectures embedded in reconfigurable devices* (CRYPTARCHI 2013), Jun. 2013

[23] J.P. Kaps, "Flexible Open-source workBench fOr Side-channel analysis (FOBOS)," Oct. 25, 2016, Internet: https://cryptography.gmu.edu/fobos/

[24] T. Schneider, A. Moradi and T. Güneysu, Arithmetic Addition over Boolean Masking," *Applied Cryptography and Network Security: 13th International Conference*, ACNS 2015, New York, Jun. 2-5, 2015, pp. 559-578.

[25] P. Kogge and H. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations". *IEEE Transactions on Computers*, 1973, *C-22*, pp. 783-791.

[26] A. Poschmann, A. Moradi, K. Khoo, C. Lim, H. Wang and S. Ling, "Side-Channel Resistant Crypto for Less than 2,300 GE," *Journal of Cryptology*, 2011, vol. 24, pp. 322-345.

[27] S. Kutzner, P. Nguyen, A. Poschmann and H. Wang, "On 3-Share Threshold Implementations for 4-Bit S-boxes," *Constructive Side-Channel Analysis and Secure Design: 4th International Workshop*, COSADE 2013, Paris, France, Mar. 6-8, 2013, Revised Selected Papers, 2013, pp. 99-113.

[28] M. Rivain and E. Prouff, "Provably Secure Higher-Order Masking of AES," *12th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES 2010, Santa Barbara, USA, Aug. 17-20, 2010, LNCS, vol. 6225, pp 413-427.

[29] F. Farahmand, A. Ferozpuri, W. Diehl and K. Gaj, "Minerva: Automated Hardware Optimization Tool," *2017 International Conference on ReConFigurable Computing and FPGAs* (ReConFig), Cancun, Dec. 2017.

[30] A. Aysu, E. Gulcan and P. Schaumont, "SIMON Says: Break Area Records of Block Ciphers on FPGAs," in *IEEE Embedded Systems Letters*, vol. 6, Jun. 2014, pp. 37-40.

[31] P. Yalla and J. P. Kaps, "Lightweight Cryptography for FPGAs," *2009 International Conference on Reconfigurable Computing and FPGAs,* Quintana Roo, Mexico, 2009, pp. 225-230.

[32] A. Shahverdi, M. Taha and T. Eisenbarth, "Lightweight Side Channel Resistance: Threshold Implementations of Simon," in *IEEE Transactions on Computers*, vol. 66, Apr. 1 2017, pp. 661-671.

[33] O. Petura, U. Mureddu, N. Bochard, V. Fischer and L. Bossuet, "A survey of AIS-20/31 compliant TRNG cores suitable for FPGA devices," *2016 26th International Conference on Field Programmable Logic and Applications* (FPL), Lausanne, 2016, pp. 1-10.