A General Degenerate Grouping Power Attack with Specific Application to SIMON and SPECK

Steven Cavanaugh

Internet Infrastructure Services Corporation
stevencavanaugh@iis-corp.com

Abstract. A Degenerate Grouping Power Attack (DGPA) is a type of Partitioning Power Analysis (PPA) used to extract secret keys from the power sidechannel signal of an encryption algorithm running on a device along with some known and varying information such as the associated plaintext or ciphertext associated with each encryption. The DGPA is applied to SIMON and SPECK implementations on MSP430, PIC16F, and Spartan 6 platforms in this work. While keys are successfully recovered from unprotected implementations, guidance is given on a minimum number of rounds, d, to perform per clock cycle in FPGAs and ASICs as to mitigate against such attacks for a deployment dependent maximum quantity of data which is to be encrypted with a given key. On the Spartan 6, full key recovery of SIMON 64/128 $d \leq 4$ and SPECK 64/128 $d \leq 3$ is trivially achieved in seconds with no more than one million random plaintexts, requiring the use of larger d for most implementations. The amount of work to recover a key as a function of the amount of collected data encrypted with that key is explored. To ensure security when performing most modes of block cipher operation with an algorithm having block size $2n$, a particular key should be used to perform no more than 2^n encryptions. A feasible key recovery requiring less than 80-bits of work and data from less than 2^{32} encryptions is excluded for SIMON 64/128 implementations having $d \geq 9$ and for SPECK 64/128 implementations having $d \geq 5$. The DGPA attack method is demonstrated to succeed against a limited data set consisting of one power sample per device clock cycle against a specifically targeted instruction. This provides a basis for a low power field deployed power side channel signal capture hardware for embedded key recovery and exfiltration.

1 Introduction

The widely known Differential Power Attack (DPA)[1], Correlation Power Attack (CPA)[2], and variants thereof have been shown to be effective in secret key recovery in encryption algorithms making use of power consumption data along with some known information about the data being processed. Such methods and variants have all been shown to be related to a general Partitioning Power Analysis (PPA)[3] which groups captured data according to an expected amount of power consumption based on the known values of data being processed.

The approach of a degenerate grouping power analysis (DGPA) is presented here which builds upon [4] and in the simplest form reduces to a 2-bit all or nothing (AON) PPA. When applied specifically to the SIMON and SPECK lightweight encryption algorithms, the DGPA method leverages properties of those algorithms to provide enhanced key recovery potential through the use of coupled key relationships between the 2 bits being examined.

This paper will explore the general approach of the DGPA and the specific application of DGPA to implementations of SIMON and SPECK in the MSP430 and PIC16F microcontrollers and to partially unrolled round-based (PUR) implementations of multiple depths on the Spartan 6 FPGA. The results extend to other platforms which exhibit different power consumption characteristics between transitioning and constant register states. Suggestions

This research was funded by the National Security Agency.

The author grants IACR a non-exclusive and irrevocable license to distribute the article under the CC BY-NC (Creative Commons Attribution-Noncommercial) license.

on minimizing the success of an attack on PUR implementations by computing a minimum number of rounds per clock are presented, an approach which has previously demonstrated resistance to a power analysis of DES implementations[5]. The frequency at which a key should be changed in order to maintain an minimum level of protection depending on the number of unrolled rounds computed per cycle is explored. A random number generator is not used in the PUR implementations, and these mitigation techniques are not subject to patent issues as is the case for masking techniques. The DGPA is shown to perform adequately using a single power measurement per clock cycle with a computational complexity scaling linearly in the number of captured encryptions, supporting the possibility of future work on lightweight embedded and field deployed power side channel attack systems.

This work only considers an attack using some known values associated with captured power data to extract an unknown key value. Other possible attacks requiring device characterization which obtain key properties through key manipulation steps in the absence of some known data component are not considered.

2 DGPA

The Degenerate Group Power Analysis (DGPA) examines $\delta \geq 2$ bit transitions in registers, logic, or other components simultaneously and groups power data according to the expected amount of power consumption due to the machine state transition. The attack requires the use of power consumption data which is associated with some known varying values, v, such as plaintext or ciphertext and will recover some set of unknown values, κ, such as encryption keys.

The meaning of the actual unknown values that are recovered are dependent on the algorithm being attacked and the specific application of the DGPA technique, and may be key bits, a linear combination of key bits, or some other nonlinear combination of key bits and other known values. For this discussion, the unknown values are termed key bits independent of the actual meaning. Operations in $GF(2)$ over single bits are denoted as $+$ for addition and $\cdot$ for multiplication.

The value of the j^{th} bit in the device at time τ, b^j_τ, can be described as a linear combination of the known quantities, v^j_τ, and the unknown quantities, κ^j_τ,

$$b^j_\tau = \kappa^j_\tau + v^j_\tau \tag{1}$$

The Hamming distance, β, describing the number of bits which change between two time adjacent states is given for a single bit by

$$\begin{aligned} \beta^j_{\tau,\tau+\Delta\tau} &= b^j_\tau + b^j_{\tau+\Delta\tau} \\ &\in \{0,1\} \end{aligned} \tag{2}$$

where $\Delta\tau$ is an offset before or after τ and represents an interval where the state of b^j remains unchanged.

In the DGPA $\delta = 2$ bit case with the unknown key values only appearing as a linear contribution, the general equations describing the two bits, $j = 0, 1$, simultaneously examined are

$$\begin{aligned} b^0_\tau &= \kappa^0_\tau + v^0_\tau \\ b^1_{\tau'} &= \kappa^1_{\tau'} + v^1_{\tau'} \end{aligned} \tag{3}$$

Note that the times at which the two bits are examined may differ for each bit.

As both $\kappa^0_\tau, \kappa^1_{\tau'}$ may actually contain multiple unknown value contributions depending on the algorithm in question, it is possible to define κ_a as a shared component and κ_b, κ_c as unique components between κ^0_τ and $\kappa^1_{\tau'}$ where $\kappa^0_\tau = \kappa_a + \kappa_b$ and $\kappa^1_{\tau'} = \kappa_a + \kappa_c$, giving

$$\begin{aligned} b^0_\tau &= \kappa_a + \kappa_b + v^0_\tau \\ b^1_{\tau'} &= \kappa_a + \kappa_c + v^1_{\tau'} \end{aligned} \tag{4}$$

This general case reduces to specific cases where one or more of the $\kappa_a, \kappa_b, \kappa_c$ may be omitted depending on the algorithm being attacked. The expected Hamming distance describing the state change of both bits is then $\beta^{0,1}_{\tau,\tau+\Delta\tau,\tau',\tau'+\Delta\tau'} = \beta^0_{\tau,\tau+\Delta\tau} + \beta^1_{\tau',\tau'+\Delta\tau'} \in \{0,1,2\}$.

The possible key hypothesis values for DGPA $\delta = 2$ are $\kappa = (\kappa_a, \kappa_b, \kappa_c) \in \mathbf{K} \equiv GF(2)^3$ with the actual correct key value denoted $k = (k_a, k_b, k_c)$. For each $\kappa \in \mathbf{K}$, a single power measurement for the specific transition time $\tau, \tau + \Delta\tau, \tau', \tau' + \Delta\tau'$ is grouped in G^κ_β for a given key hypothesis κ and resulting Hamming distance β. In cases where $\tau \neq \tau'$ and the transition occurs on two clock cycles, the single power measurement is the sum of the power consumed on each of the two transitions. The time indices used in the grouping are omitted in the group notation but their contribution to the group generation process is understood. Note that for degenerate group cases where the same β is achieved with different bit contributions, the contributions are noted in the group subscript as (b_0, b_1). Here, the case of $\beta^{0,1} = 1$ corresponds to either $\beta^0 = 1, \beta^1 = 0$ or $\beta^0 = 0, \beta^1 = 1$, forming the degenerate groups $G^\kappa_{(1,0)}$ or $G^\kappa_{(0,1)}$, respectively, which are indistinguishable in the expected power consumption.

The metric, t, follows the definition of the t-test and is used to describe the DGPA $\delta = 2$ as a PPA with $\alpha_0 = -1, \alpha_1 = 0, \alpha_2 = 1$ including a special normalization factor of

$$t^\kappa_{i,j} = \frac{s^\kappa_i - s^\kappa_j}{\sqrt{\frac{\sigma^{\kappa\,2}_i}{N_i} + \frac{\sigma^{\kappa\,2}_j}{N_j}}} \tag{5}$$

where $s^\kappa_i, \sigma^\kappa_i$ are the average and standard deviation of the single power measurement in group G^κ_i having N_i entries. The t-test method has been used in template attacks[6][7] and as a general leakage test[8] but is typically not used as the metric in PPA variants. It does provide a statistical measure as to the strength of the assertion of group ordering.

In the linear model required for CPA or multibit DPA[2], there is an assumption that the number of simultaneous bit transitions is linearly proportional to the power consumption, an assumption that is not required of the two bit all or nothing approach, thereby also avoiding the per bit characterization as required in [9]. When examining the groups the only requirement is that $s^k_{1,1} > s^k_{1,0} > s^k_{0,0}$ and $s^k_{1,1} > s^k_{0,1} > s^k_{0,0}$ and $s^k_{1,0} \approx s^k_{0,1}$.

The DGPA examines the value of $t^\kappa_{2,0}$ for each key hypothesis κ, selecting the largest t as the one corresponding to the most probable correct key hypothesis, $k^* = (k^*_a, k^*_b, k^*_c)$. The value of t is a statistical measure indicating the strength of the assertion and can be used to sort the various recovered key bits to perform targeted brute force permutations over the weakest recovered bits when a fully recovered key has not been found. From the requirement on s it is seen that the correct key yields the relationships of $t^k_{2,0} = -t^k_{0,2}, t^k_{2,0} > 0$, $t^k_{(1,0),(0,1)} = t^k_{(0,1),(1,0)} \approx 0$. A statistical fluctuation may produce a $s^k_1 > s^k_2$ resulting in a $t^k_1 > t^k_2$ and a non complete key recovery with $k^* \neq k$. However, a fluctuation is much less likely to produce a $t^k_0 > t^k_2$. Under certain conditions, an examination of the degenerate G^κ_1 groups may reveal exploitable asymmetric properties in the event of signal fluctuations.

2.1 Special Cases

There are a number of special cases for the $\delta = 2$ DGPA depending on the characteristics of the particular algorithms being attacked where one or more of the $\kappa_a, \kappa_b, \kappa_c$ may be omitted.

Single Bit PPA In the case of the unknown κ_b absent a κ_a, κ_c, the form of the two bits examined is

$$\begin{aligned} b^0_\tau &= \kappa_b + v^0_\tau \\ b^1_{\tau'} &= v^1_{\tau'} \end{aligned} \tag{6}$$

which reduces to a single bit PPA.

Two Bit AON PPA In the case of the unknown κ_b, κ_c absent a κ_a, the form of the two bits examined is

$$\begin{aligned} b_\tau^0 &= \kappa_b + v_\tau^0 \\ b_{\tau'}^1 &= \kappa_c + v_{\tau'}^1 \end{aligned} \tag{7}$$

which reduces to a two bit all or nothing PPA. A fully incorrect recovered key produces a negated t, so it is substantially more likely that an incorrect key pair recovery will recover at least one of the key bits correctly. The fact that the fully incorrect key corresponds to $G_2^k = G_0^{\overline{k}}$ is defined as a normal hierarchy. A statistical fluctuation as a result of $s_1 > s_2$ is equally as likely to recover $(k_b^*, k_c^*) \in \{(k_b, \overline{k_c}), (k_c, \overline{k_b})\}$.

Identifying Unrecoverable Key Bits via G_1 In the case of the unknown κ_a absent a κ_b, κ_c, the form of the two bits examined is

$$\begin{aligned} b_\tau^0 &= \kappa_a + v_\tau^0 \\ b_{\tau'}^1 &= \kappa_a + v_{\tau'}^1 \end{aligned} \tag{8}$$

The single shared unknown variable provides a single bit result which may be enhanced in accuracy by using the two different measurement points. However, when allowing the unknown component in each bit value to take on different values such as

$$\begin{aligned} b_\tau^0 &= \kappa_b + v_\tau^0 \\ b_{\tau'}^1 &= \kappa_c + v_{\tau'}^1 \end{aligned} \tag{9}$$

where the recovery of $\kappa_b = \kappa_c$ indicates a concurrence in the value of the shared unknown quantity. As the equations are overly defined, a fluctuation yielding $t_{1,0} > t_{2,0}$ would produce an identifiable inconsistent state in a G_1 degenerate group which indicates a definitive inability to recover the key state. The identification of bits which cannot be determined from the degenerate G_1 group is the first application of the DGPA.

Coupled Equations, Inverted Hierarchy In the case of the unknown κ_a, κ_b absent a κ_c, the form of the two bits examined is

$$\begin{aligned} b_\tau^0 &= \kappa_a + \kappa_b + v_\tau^0 \\ b_{\tau'}^1 &= \kappa_a + v_{\tau'}^1 \end{aligned} \tag{10}$$

Here the two bits are coupled by the unknown κ_a. Note that the groups formed under a particular key hypothesis $\kappa = (\kappa_a, \kappa_b)$ are related to the other key hypothesis with $G_{1,1}^{\kappa_a,\kappa_b} = G_{0,0}^{\overline{\kappa_a},\kappa_b} = G_{0,1}^{\kappa_a,\overline{\kappa_b}} = G_{1,0}^{\overline{\kappa_a},\overline{\kappa_b}}$ and $G_{0,0}^{\kappa_a,\kappa_b} = G_{1,1}^{\overline{\kappa_a},\kappa_b} = G_{1,0}^{\kappa_a,\overline{\kappa_b}} = G_{0,1}^{\overline{\kappa_a},\overline{\kappa_b}}$ If the recovered key is partially correct as $k^* = (k_a, \overline{k_b})$, then $t_{2,0}^{k_a,\overline{k_b}} \approx 0$. Similarly, a fully incorrect key yields $t_{2,0}^{\overline{k_a},\overline{k_b}} \approx 0$. If instead the partially correct recovered key is $k^* = (\overline{k_a}, k_b)$ then $t_{2,0}^{\overline{k_a},k_b} \approx -t_{2,0}^{k_a,k_b}$. Since the largest t has been selected to obtain the most probable correct key, it is seen that a partially recovered key with an incorrect k_b is as likely as an fully incorrect recovered key, both of which are more likely than recovering an incorrect k_a alone. The fact that the the fully incorrect key appears other than in G_0 results in a inverted hierarchy relationship where $G_2^{k_a,k_b} = G_{(0,1)}^{\overline{k_a},\overline{k_b}}$.

The probability of the G_1 degenerate groups giving rise to a $t_{1,0} > t_{2,0}$ is equal for either $t_{(1,0),(0,1)}$ and $t_{(0,1),(1,0)}$, both of which include a wrong κ_a. Since only one of the degenerate groups contains a wrong κ_b, the probability of a wrong κ_b is half that of a wrong κ_a when a degenerate G_1 group is selected instead of the proper G_2 group. As the probability of selecting the wrong κ_b through a incorrect selection of the G_0 group is much less than selecting the G_1 group, the inverted hierarchy can be said to be biased in selecting a correct κ_b over κ_a with almost a 2:1 enhancement. As such, coupled equations with inverted hierarchy provide the second application of the DGPA and should be used when the algorithm under attack allows such application.

Leveraging Underdetermined Degenerate G_2 In the case of the unknown $\kappa_a, \kappa_b, \kappa_c$, the form of the two bits examined is the general case

$$\begin{aligned} b_\tau^0 &= \kappa_a + \kappa_b + v_\tau^0 \\ b_{\tau'}^1 &= \kappa_a + \kappa_c + v_{\tau'}^1 \end{aligned} \tag{11}$$

The κ cannot be definitely determined, as $G_2^{k_a,k_b,k_c} = G_2^{\overline{k_a},\overline{k_b},\overline{k_c}}$. However, the result of the attack is an identification of the relationship between the k_a, k_b, k_c, where the fixing of one of the values along with the known relationship between the values reveals the state of the other two. The use of the set of G_2^κ which are correctly identified is the third application of the DGPA whereby a limited attack targeting a particular G_2^κ provides all other G_2 at no additional cost. When there are multiple degenerate G_2, the correct G_2 is denoted as G_C.

3 Experimental Method

A custom circuit board was created for each of the Spartan 6 XC6SLX4, MSP430G2553, and PIC16F1618 to provide data, clock, and power connections to the device under test. In all cases the supply power (V_{INT} for the Spartan) is measured with a two stage instrumentation amplifier[10] comprising three LM6181 Operational Amplifiers providing an overall effective gain of 105× the current signal when driven through a 50Ω terminated cable. The voltage drop across a sense resistor of 10Ω provides input to a first stage with gain of 21× (with $100k\Omega$ and $10k\Omega$ resistors) and with unity gain in the second stage (with $1k\Omega$ resistors).

Separate power is provided for $V_{IO} = 2.5V$, $V_{INT} = 1.2V$, $V_{AUX} = 3.3V$ with three Keithley 2304A power supplies for the Spartan, and with $V = 3.5V$ with a single supply for the microcontrollers. A HP6624A power supply is used to provide $\pm 15V$ for the LM6181. Data is captured on a Agilent MSO9404A digitizing oscilloscope in segmented mode with external device clock provided from a HP 8112A. A NI PXIe-8135 controller in a NI PXIe-1075 chassis runs the test code and provides GPIB test equipment control. Test data and experiment configuration communication with the device under test is achieved with a NI PXI-6509 Digital I/O module which provides 5V TTL signals which are translated as required with an external circuit.

Device test code and configurations are created with the necessary ability to handle the external data transfer of plaintext and ciphertext with the control machine. Random plaintext is created on the device by using the ciphertext output of an encryption as the random plaintext of the next encryption. Encryptions between fixed and random plaintexts are interleaved. Test code for the microcontroller includes code which provides a marker in the power signal to aid in synchronization. Recorded traces are processed offline to merge multiple oscilloscope files and to extract the power signal pulse amplitude as a single value per clock pulse. Average traces are computed offline for clock signals corresponding to the same operation in the test separately for the fixed and random plaintext samples.

The fixed plaintext and key are loaded at the start of the test and are taken as the published test values[11] unless stated otherwise. Due to limited space on the device from constraints outside of this study, the round keys used for the four round MSP SPECK 64/128 test are those obtained from the $m = 3$ key expansion and SPECK 64/96 test key and the Spartan SIMON and SPECK 128/256 test cases derive round keys from the $m = 2$ key expansion and SIMON/SPECK 128/128 test key. These substitutions do not effect this study as the round function under test is the same for all m, the number of rounds examined is less than T, and the constant power consumption of the key schedule algorithm cancels when computing power differences. The pseudo random plaintext is initialized as zero and the ciphertext resulting from that encryption is taken as the pseudo random plaintext input for the next encryption. Unless stated otherwise, all results are generated with 50k events each of fixed and random plaintext.

A *trace* refers to the raw measurement of the power consumption of a device as made by an oscilloscope or other signal acquisition hardware with one data point per time sample

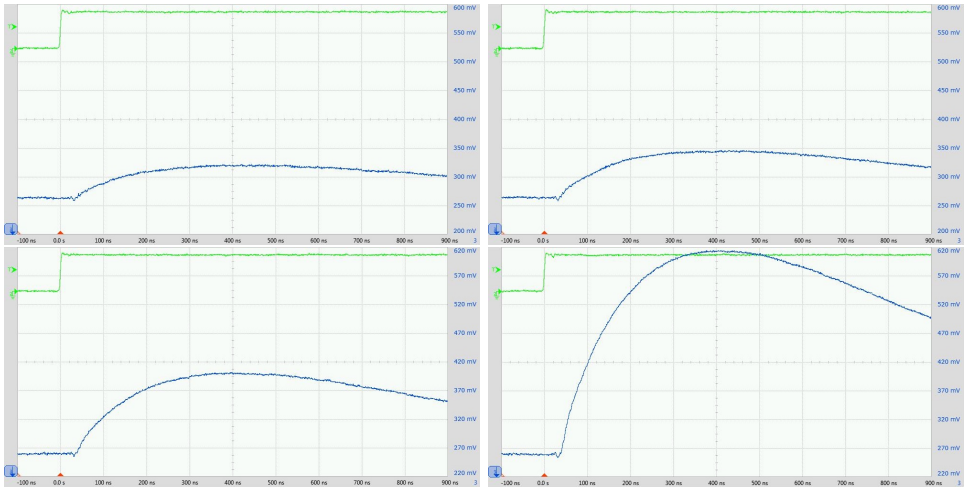


Fig. 1: A picture of a single trace captured for the second cycle of the encryption for SIMON (top) and SPECK (bottom) for $d = 4$ configurations of 64/128 (left) and 128/128 (right). The clock pulse is shown in green (upper trace in each picture) and the amplified V_{INT} signal is shown in blue. Note how the power consumption with increasing n is substantially greater for SPECK than for SIMON due to the cascading effects of the addition carry. The vertical range showing the raw power measurement is $400mV$ and the horizontal range is $1\mu s$.

of the acquisition system. The timing of the data acquisition is usually not synchronized with the clock of the device under attack and as a result, multiple traces of the same device activity will not precisely align. The use of a trace can be computationally intensive due to the number of traces and the number of points in each trace being processed. The signal of interest is generally expected to occur at a single instruction or clock cycle in an algorithm, and once that location is known, only the power data corresponding to that exact point of processing must be examined. The preprocessing of the recorded trace data extracts a single power measurement value per clock cycle. For the Spartan 6, the power peak is measured as the average of the 21 sample points centered at the 50% point of the capture window which roughly corresponds to the signal peak, with example signals given in figure 1. For the MSP and PIC, the power peak is measured as the maximum sample point in the capture window after the rising clock edge. The resulting power measurement is the power peak subtracted from the baseline which is calculated as the average of the first 5% of the capture window prior to the rising clock edge. While the tests described use an external clock trigger to record one segment per clock, the successful recovery of the power consumption peaks associated with an internal clock using a single recorded trace was also studied and found to produce equivalent results.

Specially designed tests were carried out to exercise and characterize the behavior of the registers in each device type. Directional asymmetry between a register state transition flip up ($0 \rightarrow 1$) and flip down ($1 \rightarrow 0$) was examined and is presented in table 1.

4 SIMON and SPECK

SIMON and SPECK are both lightweight block ciphers[11] which are respectively optimized for hardware and software implementations. The block size is $2n$, the number of n-bit key words is m, and the number of rounds is T. The parameters vary for different implementations. The key of size nm bits is expanded to obtain T round keys each of n bits. The key schedule algorithm is unrolled with the corresponding encryption rounds on the Spartan and round keys are expanded into FLASH for the MSP and PIC prior to encryption.

Device	V_{CC}	$0 \to 1$ (u) (V/bit)	$1 \to 0$ (d) (V/bit)	(d) Asymmetry $\frac{u}{d} - 1$
Spartan 6	1.4	1.74E−5 ± 3.62E−8	1.70E−5 ± 3.64E−8	2%
MSP430	3.5	6.73E−5 ± 1.84E−6	5.23E−5 ± 2.01E−6	29%
PIC16F (SRAM read)	3.5	7.79E−5 ± 4.27E−5	1.52E−6 ± 6.57E−6	5034%
PIC16F (Data into ALU)	3.5	3.40E−5 ± 6.72E−6	1.08E−4 ± 1.48E−5	-68%
PIC16F (ALU onto bus)	3.5	2.32E−4 ± 1.82E−5	1.80E−4 ± 5.48E−6	29%
PIC16F (SRAM write)	3.5	4.96E−5 ± 5.41E−6	4.79E−5 ± 1.10E−5	4%

Table 1: Experiments exercising different number of register bit transitions, x, revealed linear dependence on the signal power, y, presented as the baseline subtracted voltage peak not including the 105× gain of the instrumentation amplifier. The linear fit parameter p_1 is presented as V/bit for the fit $y = p_0 + p_1 x$ where the arbitrary p_0 offset is not consequential. No asymmetry is seen in the Spartan 6, while a similar 29% asymmetry is seen in both the MSP430 and PIC16F on the register state change.

The block of size $2n$ is divided into left and right partitions, $\mathbf{X}, \mathbf{Y}$ which are the upper and lower halves of the block. The input to round i is $\mathbf{X}_i, \mathbf{Y}_i$, the output is $\mathbf{X}_{i+1}, \mathbf{Y}_{i+1}$, and the round key used in that round is k_i. The notation of $\mathbf{X}^\lambda$ indicates a λ bit circular shift to the left. The n-bit word $\mathbf{X}$ has the bit value X^j at index j, with $(\mathbf{X}^\lambda)^j = X^{j+\lambda}$.

Operations over single bits in $GF(2)$ are denoted as + for addition and · for multiplication. When operating over n-bit words denoted by bolded letters, the notation of bitwise-addition $\oplus$, bitwise-multiplication &, and addition with carry + is used.

The SIMON round function makes use entirely of bitwise operations and is

$$\begin{aligned} \mathbf{X}_{i+1} &= \mathbf{Y}_i \oplus \mathbf{X}_i^2 \oplus (\mathbf{X}_i^1 \& \mathbf{X}_i^8) \oplus \mathbf{k}_i \\ \mathbf{Y}_{i+1} &= \mathbf{X}_i \end{aligned} \tag{12}$$

The SIMON keyless round function is defined as $\mathbf{A}_i = \mathbf{Y}_i \oplus \mathbf{X}_i^2 \oplus (\mathbf{X}_i^1 \& \mathbf{X}_i^8)$.

The SPECK round function makes use of arithmetic addition and is specified with parameters $\alpha = -8, \beta = 3$ for all but the simplest configurations and is

$$\begin{aligned} \mathbf{X}_{i+1} &= (\mathbf{X}_i^\alpha + \mathbf{Y}_i) \oplus \mathbf{k}_i \\ \mathbf{Y}_{i+1} &= \mathbf{Y}_i^\beta \oplus (\mathbf{X}_i^\alpha + \mathbf{Y}_i) \oplus \mathbf{k}_i \end{aligned} \tag{13}$$

The SPECK keyless round results are given as $\mathbf{W}_{i+1} = \mathbf{X}_{i+1} \oplus \mathbf{k}_i$ and $\mathbf{Z}_{i+1} = \mathbf{Y}_{i+1} \oplus \mathbf{k}_i$. A bit-wise representation of SPECK replaces the addition with an explicit carry term contribution

$$\begin{aligned} X_{i+1}^j &= X_i^{\alpha+j} + Y_i^j + C_i^j + k_i^j \\ Y_{i+1}^j &= X_{i+1}^j + Y_i^{\beta+j} \end{aligned} \tag{14}$$

where

$$\begin{aligned} C_i^{j+1} &= X_i^{\alpha+j} \cdot Y_i^j + (X_i^{\alpha+j} + Y_i^j) \cdot C_i^j \\ C_i^0 &= 0 \end{aligned} \tag{15}$$

The implementations of each algorithm tested in this paper follow from the suggested ATmega implementations[12], with adjustments made only to account for the 16 bit registers in the MSP430, and the various different available commands in both. The Spartan 6 implementations utilize a single $2n$ bit register comprised of Flip Flop elements with a LUT5 performing the round function for each bit for SIMON and the CARRY chain bordered by LUTs performing the round function for SPECK. The register is specially designed and is bordered by LUTs which control the data source and sink of the register, loading plaintext data from a bus before encryption begins and taking data from the round logic during encryption, and placing ciphertext onto the bus only once encryption is complete and onto

the round logic during encryption. Also, as the data can be loaded from a bus size less than $2n$, the data is not placed on the round logic until all data is loaded into the registers to prevent an attack on only a part of the round logic. For both algorithms, the combined LUT/CARRY components are referred to as the LUT tree. The logic of the LUT tree may be repeated so that d rounds are computed per clock cycle. During encryption, a single clock cycle stores the output of the LUT tree into the edge triggered flip flop register which immediately changes the value which is present on the inputs to the LUT tree. The signal that is generated is from the overwriting of the round i result with the round $i + d$ result, while the LUT tree changes state from that with the round i input to that with the round $i + d$ input, resulting in the $i + 2d$ round result being present at the LUT tree output.

The number of bits computed per clock cycle, ν, is fixed at the register width of 8 and 16 for the PIC and MSP respectively, and is set at $\nu = n$ for the Spartan implementations examined.

Data is recorded for the first four rounds of encryption on the PIC and MSP and for the first $11d$ rounds on the Spartan. The choice of 4 rounds for the microcontrollers was made to limit the data set size as used in a different study. The choice of $11d$ rounds is only an artifact of initial $d = 4, T = 44$ SIMON 64/128 testing, with later implementations using the same $11d$ for comparison purposes. As the correct key is known, each round can be examined as a first round with the appropriate expanded key offset with the correct key and plaintext used to provide the correct input to the round under test.

5 Attacks on Single Instructions

On both studied microcontrollers having register widths less than n, only a part of the block is processed in any given clock cycle. Examining two bits of any particular single clock cycle, the signal always takes the form of noncoupled normally ordered groups and gives a result equivalent to what would be achieved with 2-bit AON PPA. The ability to identify the round looping, as shown in figure 2, is essential to targeting specific instructions. The combination of power signals from different clock cycles could be used to leverage the specific aspects of the DGPA approach, but are not studied here.

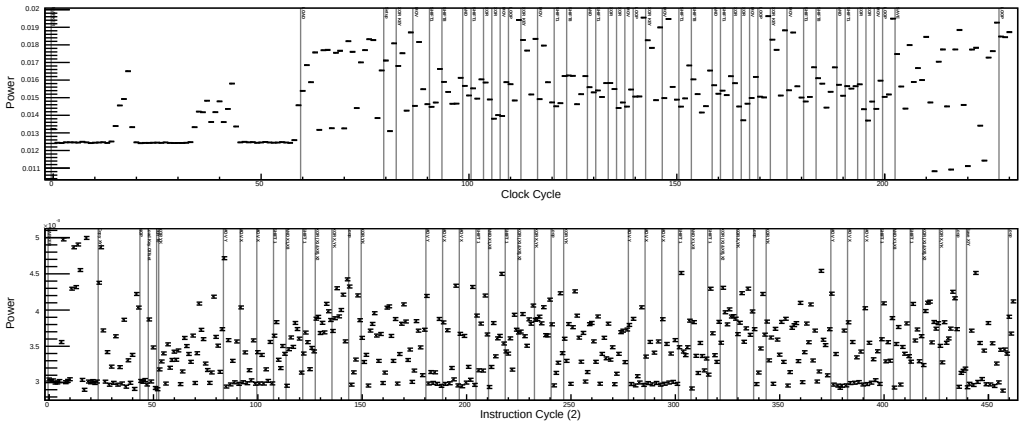
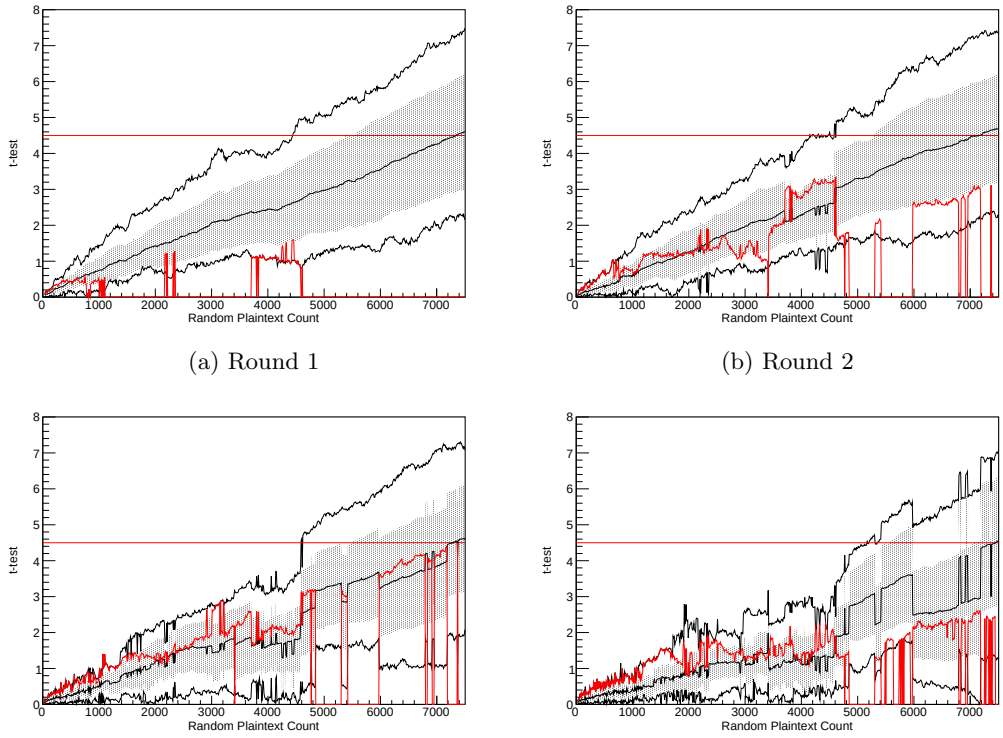


Fig. 2: The raw single power signal (y-axis) per clock cycle (x-axis) for the encryption of fixed plaintext for four rounds of standard SIMON 64/128 for the MSP430 (top) and the third clock cycle (ALU latch result to data bus) for the PIC16F (bottom). Clear patterns are seen and individual instructions can be identified and targeted in the power analysis, as denoted by the labeled vertical lines.

MSP The MSP430 16-bit register overwrite is attacked for SIMON 64/128 on the XOR of $Y \oplus K$ with $(X^1 \& X^8) \oplus X^2$ to fully recover the first four round keys including propagating error in under 5,500 plaintexts. A similar attack on SPECK 64/128 on the XOR of $(X^8 + Y) \oplus K$ with Y^3 yields the first round key completely in under 1k plaintexts and the first four round keys including propagating in about 5k plaintexts. The t-test corresponding to the recovery is presented in figure 3.



(a) Round 1

(b) Round 2

(c) Round 3

(d) Round 4

Fig. 3: The t-test for the incremental degenerate attack on four rounds of SPECK 64/128 on the MSP430, where a round key is recovered using the keys found for previous rounds. The minimum, average, and maximum t-test of correctly identified key bits are given in black, with the 1σ from average as the shaded area. The red line is the maximum t-test of all incorrectly recovered key bits. Note that full key recovery across four rounds is achieved with approximately 5000 plaintexts, a point at which the minimum t-test value is still $\approx 1\sigma$. This early recovery of the complete first four round keys is followed by partially incorrect recovery for greater plaintext count due to statistical fluctuations. The maximum incorrect t-test value remains near 2σ as the average correct t-test value increases beyond that threshold for rounds 2 and 4, indicating that the t-test value serves as a measure of confidence on the correctness of the recovered key. This is not seen in the round 3 recovery due to the propagating error introduced by incorrect recovery in round 2 and not due to the recovery attempt in round 3 directly.

PIC On the PIC the four clock cycles per instruction cycle are clearly separable in the data and the attack is found to work on the clock cycles corresponding to the loading of the W register in the second cycle and on the placing of the ALU output value on the data bus in the third cycle. The reading and writing to the SRAM registers does not produce a large signal as these registers are not the typical file register design and are not constantly amplified. A four round key recovery with propagating errors of SPECK 64/128 is achieved with 1.3k random plaintexts by examining the power of an offset increment command which reveals the signal on the ALU latch which previously computed $\mathbf{k} \oplus (\mathbf{X}^{-8} + \mathbf{Y})$. The same attack on the offset increment command in a SIMON 64/128 implementation fully recovers four round keys with propagating errors with 930 random plaintexts from the signal on the ALU latch which previously computed $\mathbf{k} \oplus \mathbf{Y}$.

6 Simple Attacks on Spartan 6 SIMON/SPECK d=1,2

Average key recovery over the eleven rounds of Spartan 6 sample data was examined for different attack methods. The best performing attacks operating on the partially unrolled rounds (PUR) Spartan implementations of $d = 1$ have only one clock cycle to examine per round and recover the key completely with 14.1k, 12.5k , 16.6k, 34.4k random plaintexts for SIMON 64, SIMON 128, SPECK 64, and SPECK 128 as shown in table 2. A simple approach for recovery of SIMON $d = 2$ is also feasible as an examination of the $\mathbf{Y}_2 \rightarrow \mathbf{Y}_0$ overwrite can be used to find $\mathbf{k}_0$ which is then used to recover $\mathbf{k}_1$ in the $\mathbf{X}_2 \rightarrow \mathbf{X}_0$ overwrite. The average rate of key recovery as a function of the number of plaintexts, x, is described by the equation $p_0 + p_1/(p_2 + x)$, where the parameters are obtained from fits to the data for the various methods and are presented in table 3.

				Fully Understood		Close ($> n - \frac{1}{2}$)	Max Recovery	
	n	d	Type	Samples	Undetermined	Recovery	Avg Bits	Samples
SIMON	32	1	AON X	14100	0	3060	32	14100
	32	2	AON $(Y_2 \rightarrow Y_0) \rightarrow k_0$	-	-	28830	31.55	28830
	32	2	AON $(X_2 \rightarrow X_0) \rightarrow k_1$	-	-	-	31.18	44990
	32	2	Coupled $(k_0, k_1) \rightarrow k_0$	-	-	28510	31.55	28510
	32	2	Coupled $(k_0, k_1) \rightarrow k_1$	-	-	-	31.18	45030
	64	1	AON X	12500	0	8100	64	12500
	64	1	AON Y	31240	0	25790	64	31240
	64	1	1-bit PPA X	12500	0	8090	64	12500
	64	1	4-bit PPA X	23120	0	14970	64	23120
SPECK	32	1	AON X	-	-	16000	31.82	18290
	32	1	AON Y	16560	0	9400	32	16560
	32	1	G_1	4260	3.455	16000	31.82	18280
	32	2	$d = 2$ Single Pass	-	-	30560	31.82	41700
	64	1	AON X	41560	0	17760	64	41560
	64	1	AON Y	-	-	14440	63.91	32740
	64	1	G_1	14550	1.000	19800	64	34380

Table 2: The number of random plaintext encryption samples required for key recovery using various attack methods are presented for Spartan 6 implementations of SIMON and SPECK with block size $2n$ computing d rounds per clock cycle. Results represent the average behavior obtained when treating the first 11 clock cycles of each encryption implementation as a distinct experiment with known random plaintext input. AON X/Y examines two bits of either X or Y which are uncoupled and which reduce to the 2-bit AON PPA case. G_1 is the DGPA overly defined case. Fully Understood results represent the first time all bits are understood (either known or identified as undetermined). SIMON d=1 AON Y attacks the Y overwrite one cycle after the round result is stored in X. SPECK d=2 Single Pass is not a simple attack, but is included for completeness and is described in section 9.

	n	d	Type	p_0	p_1	p_2	$R_x^{\frac{3}{4}n}$	R_x^{n-1}
MSP SIMON	32	1	AON	34.7744	-20607.8	1108.60	804	4351
MSP SPECK	32	1	AON	32.8089	-5199.00	335.961	254	2538
PIC SIMON	32	1	AON ALU	32.4197	-1022.21	46.8146	75	673
PIC SPECK	32	1	AON ALU	32.2347	-777.991	37.5202	57	593
Spartan SIMON	32	1	AON X	32.3280	-4805.37	293.007	284	3325
	32	2	AON $(Y_2 \to Y_0) \to k_0$	31.7145	-14269.1	1059.22	790	18912
	32	2	AON $(X_2 \to X_0) \to k_1$	31.6868	-30244.1	1870.05	2065	42166
	32	2	Coupled $(k_0, k_1) \to k_0$	31.7444	-14417.1	1069.23	792	18298
	32	2	Coupled $(k_0, k_1) \to k_1$	31.7292	-30207.5	974.972	2933	40451
	64	1	AON X	64.9898	-14401.5	533.406	314	6704
	64	1	AON Y	65.7042	-63133.5	2057.36	1509	21289
	64	1	1-bit PPA X	64.5060	-12271.5	436.722	307	7712
	64	1	4-bit PPA X	64.9641	-32573.6	1275.67	644	15309
Spartan SPECK	32	1	AON X	32.5635	-20571.6	1432.45	970	11725
	32	1	AON Y	32.6074	-11707.5	726.908	633	6557
	32	1	G_1	32.5633	-20673.6	1439.76	974	11785
	32	1	G_1 Determined	32.1938	-1977.84	269.749	-28	1387
	32	2	$d = 2$ Single Pass	32.8795	-45971.1	2699.43	2478	21760
	64	1	AON X	64.9445	-37269.4	1421.49	778	17745
	64	1	AON Y	64.5371	-20488.7	717.674	521	12612
	64	1	G_1	64.9595	-37573.7	1418.26	797	17757
	64	1	G_1 Determined	64.1698	-4723.84	385.530	-93	3653

Table 3: The fit of the average key recovery rate over the 4 rounds tested for MSP and PIC and 11 rounds tested for Spartan follows the form $f(x) = p_0 + \frac{p_1}{(p_2+x)}$ for x plaintexts for the SIMON and SPECK implementations with block size $2n$ computing d rounds per clock cycle. The expected number of plaintexts required for a recovery of $\frac{3}{4}n$ and $n-1$ of the round key bits is given as $R_x^{\frac{3}{4}n}$ and R_x^{n-1} with both obtained from the fit parameters, where a lower value indicates a stronger method for the same platform and algorithm configuration. No particular enhancement is seen in the inverted hierarchy coupled approach for Spartan SIMON $d = 2$. The total number of key bits correctly identified or specifically determined to be unrecoverable by the G_1 degeneracy is reported as G_1 determined. For the smallest SPECK $d = 1$ sample of 10 random plaintexts, round key recovery already exceeds $\frac{3}{4}n$ bits with actual results of 25 and 52 bits recovered or determined to be unrecoverable for SPECK 64 and SPECK 128, respectively. SPECK d=2 Single Pass is not a simple attack, but is included for completeness and is described in section 9.

7 Advanced Attacks on SIMON $d > 2$

The SIMON equations for $\mathbf{X}_{i+d}, \mathbf{Y}_{i+d}$ expressed in terms of the inputs $\mathbf{X}_i, \mathbf{Y}_i$ and the keys $k_i, k_{i+1}, ..., k_{i+d-1}$ always contain linear components of $\mathbf{X}_i, \mathbf{Y}_i$. The probability, p_1, that a bit j from a sample of random plaintext takes a value of 1 is $p_1(X_i^j) = p_1(Y_i^j) = p_1(X_{i+d}^j) = p_1(Y_{i+d}^j) = \frac{1}{2}$ for a sufficiently large sample. All nonlinear terms of the form $(X_j^1 \cdot X_j^8)$ then have a probability $p_1 = \frac{1}{4}$.

As implementations with larger d include more nonlinear terms in the equations for X_{i+d}, Y_{i+d}, the nonlinear terms have a larger probability of contributing to the value. Assuming that all i level terms (inputs to the cycle) only appear once each in each linear term, the probability that the entire nonlinear component of λ terms will give an overall value of

one is dependent on an odd number of active nonlinear terms and is given by

$$p_{nonlin}(\lambda) = \sum_{odd\ i}^{\lambda} \frac{\lambda!}{i!(\lambda-i)!}\left(\frac{1}{4}\right)^{i}\left(\frac{3}{4}\right)^{\lambda-i} \tag{16}$$

The linear approximation of SIMON takes a general form of

$$\begin{aligned} X_{i+d}^{j} &= k_A^j + k_B^j + f(\mathbf{X}_i, \mathbf{Y}_i)^j \\ Y_{i+d}^{j+2} &= k_A^j + g(\mathbf{X}_i, \mathbf{Y}_i)^{j+2} \end{aligned} \tag{17}$$

where k_A, k_B are linear combinations of key bits and the shifted Y_{i+d}^{j+2} term contains key bits which also appear in X_{i+d}^{j}. The index of the bit being computed is j and the round being computed is $r = i + d$. The functions f, g exclude nonlinear contributions which include key terms.

A first pass of the data is used to calculate $(\kappa^A, \kappa^B)_r^j$ as is done for $d = 1, 2$. For $d \geq 3$, $d-2$ additional passes are required to extract the key bits. In all cases, there are matching nonlinear terms between X^j, Y^{j+2} in the form of $(X_l^{q+1+j} \cdot X_l^{q+8+j})$ that were previously omitted from f, g where l depends on the level being recovered and q depends on the actual expansion which is d dependent. Those omitted terms can be used to probe the contained key information when included individually. For instance

$$\begin{aligned} X_{i+d}^{j} &= k_A^j + k_B^j + f(X_i, Y_i)^j + (X_{i+1}^{q+1+j} \cdot X_{i+1}^{q+8+j}) \\ &= k_A^j + k_B^j + f(X_i, Y_i)^j + ((k_i + A_i)^{q+1+j} \cdot X_{i+1}^{q+8+j}) \\ &\approx k_A^j + k_B^j + f(X_i, Y_i)^j + (k_i + A_i)^{q+1+j} \\ Y_{i+d}^{j+2} &= k_A^j + g(X_i, Y_i)^{j+2} + ((k_i + A_i)^{q+1+j} \cdot X_{i+1}^{q+8+j}) \\ &\approx k_A^j + g(X_i, Y_i)^{j+2} + (k_i + A_i)^{q+1+j} \end{aligned} \tag{18}$$

A similar procedure is carried out for the additional pass to recover k_i as in the first pass, where the value of A_i^j is $Y_i^j + X_i^{j+2} + (X_i^{j+1} \cdot X_i^{j+8})$. In the case where $X_{i+1}^{q+8+j} = 1$ and the remaining linear combination of non linear terms is zero then the data is properly grouped. However, approximately half of the time the assumption of $X_{i+1}^{q+8+j} = 1$ is invalid (the term is instead zero), which causes the data to be improperly grouped ($G_2 \rightleftharpoons G_0$) only when $(k_i + A_i)^{q+1+j} = 1$ which happens half of the time. So the overall probability of improperly grouping the data is $1/4$ and the sample is still biased to reveal k_i. For $i > 0$ it is necessary to use the previously found values of $k_0, ..., k_{i-1}$ to calculate A_i^j which results in a propagation of error if there are incorrect previously found k bits.

The linear approximation begins to fail as the probability of a nonlinear term contribution increases. The unknown nonlinear term contribution, shown in table 4 demonstrates how the $\mathbf{X}, \mathbf{Y}^2$ term coupling serves to prolong the usefulness of the linear approximation. The DGPA offers an enhancement over the best performing of the other 2-bit methods for most d, although it is greatest for $d = 5$. The DGPA enhancement comes from the fact that the observation of two equations requires both to be correct linear approximations with no overall nonlinear contribution. Considering the individual probabilities of the shared terms and unique terms giving rise to overall zero nonlinear contribution, the probability of that occurring is greater than two uncoupled equations containing no shared terms giving rise to overall zero nonlinear contribution. Examining less bits at once (1-bit PPA) gives the greatest probability of a correct linear approximation, although the observation of multiple bits can in practice offset that benefit with improved statistics and signal significance. An estimation of the resulting t-test for various d due to contamination of the G_2, G_0 groups is given in table 5, along with an estimation of the relative scaling of the data sample required for a given d to obtain a statistically significant result comparable to the $d = 1$ case.

d	NTC			Correct Linear Approximation Fraction						Max DGPA
	X^j	Y^j	X^j, Y^{2+j}	X^j	Y^j	X^j, Y^{2+j}	2-bit X	2-bit Y	2-bit X, Y	Enhancement
$n = 24, 32, 48, 64$										
1	0	0	0	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.0000%
2	1	0	0	0.750000	1.000000	0.750000	0.562500	1.000000	0.750000	-25.0000%
3	2	1	1	0.625000	0.750000	0.562500	0.390625	0.562500	0.468750	0.0000%
4	4	2	2	0.531250	0.625000	0.390625	0.282227	0.390625	0.332031	0.0000%
5	5	4	3	0.515625	0.531250	0.304688	0.265869	0.282227	0.273926	7.9585%
6	8	5	4	0.501953	0.515625	0.266602	0.251957	0.265869	0.258820	0.2755%
7	10	8	6	0.500488	0.501953	0.255127	0.250489	0.251957	0.251222	1.2582%
8	13	10	7	0.500061	0.500488	0.250763	0.250061	0.250489	0.250275	0.1096%
9	14	13	8	0.500031	0.500061	0.250168	0.250031	0.250061	0.250046	0.0427%
10	18	14	9	0.500002	0.500031	0.250031	0.250002	0.250031	0.250016	0.0004%

Table 4: The total unique unknown nonlinear term count (NTC) for SIMON for various number of rounds computed per cycle (d) is presented, where zeroth order nonlinear terms are not included because they are calculable and nonlinear terms which appear twice and cancel when XOR'd have been removed. The X^j, Y^{2+j} NTC column indicates the number of terms which appear in both of the coupled equations for X^j and Y^{2+j}, and the number of unique terms in the coupled equation is obtained by subtracting the terms appearing in both from the total terms in each. The fraction of random plaintext which is described correctly by the linear approximation is given for single bits of X and Y (1-bit PPA), a pair of bits from the coupled equations X^j, Y^{2+j} (DGPA), and for 2-bit PPA pairs taking bits from only X, only Y, or both X and Y.

Interestingly, this attack method is not strongly dependent on n until d is great enough that the terms which appear in the linear expansion also appear as an outer term in a non linear term (a greater n provides less interference for the same d). The method could presumably be applied to implementations which do not have fully unrolled rounds provided that the power is added for the cycles corresponding to the necessary $\mathbf{X}$ and $\mathbf{Y}$ shift. For a specific d attack, there may be multiple equation pairs that may be used for various terms to either verify key bit results or identify probable incorrect key bits. Even just recovery with a single pass of $(k_A^*, k_B^*)_r^j$ provides ample information on the relationship between key bits to provide at least one full round key for a brute force attack on the remaining $m-1$ round keys. For this reason, it is necessary to choose d such that the nonlinear term contribution is as close to half of the total contribution as possible.

The attack is tested for each of the encryption cycles independently, as the correct input to each cycle can be calculated using the known key. The actual recovery rate is shown in table 6. While the ability to recover key bits for $d \geq 5$ is dependent on the recovery of k_A^* and k_B^*, the ability to do so will be dependent on the data sample size which will need to be increased to compensate for the decreasing significance of the linear approximation. Using the sample size scaling from table 5, and the fact that 14.1k plaintexts achieved full key recovery for $d = 1, n = 32$ on the Spartan, an arbitrary requirement of preventing full key recovery with fewer than 5 million plaintext results in a minimum recommendation of $d = 5$.

The treatment given here assumes that single buffers are used for $\mathbf{X}, \mathbf{Y}$ and that $\mathbf{X}_{i+d}$ overwrites $\mathbf{X}_i$ and $\mathbf{X}_{i+d-1}$ overwrites $\mathbf{Y}_i$. This same approach holds for other implementations, including ping pong buffers or hybrid combinations providing that the initial state of the destination buffer is known to the attacker or can otherwise be determined or controlled. In the case of $\nu < n$ it is necessary to perform proper bookkeeping as the power corresponding to some $\mathbf{X}, \mathbf{Y}^2$ pairs occur on the same cycle, while in other cases the power from two cycles must be combined. In the ping pong buffer situation, if the contents of the first destination buffer are not known, then the attack must be performed on the overwrite of the first source buffer which effectively results in a $2d$ attack complexity. The only way to ensure that the first destination buffer value is not known is to load a new random value into that buffer before each encryption.

d	Components of G_2 Selection G_2	G_0	G_1	$t_{2,0}$	n_2, n_0
$n = 24, 32, 48, 64$					
1	100.0000000%	0.0000000%	0.0000000%	2.0000E0	1.0000E0
2	75.0000000%	0.0000000%	25.0000000%	1.5000E0	1.7778E0
3	56.2500000%	18.7500000%	25.0000000%	7.5000E-1	7.1111E0
4	39.0625000%	23.4375000%	37.5000000%	3.1250E-1	4.0960E1
5	30.4687500%	25.7812500%	43.7500000%	9.3750E-2	4.5511E2
6	26.6601562%	24.9023438%	48.4375000%	3.5156E-2	3.2363E3
7	25.5126953%	25.2685547%	49.2187500%	4.8828E-3	1.6777E5
8	25.0762939%	25.0213623%	49.9023438%	1.0986E-3	3.3140E6
9	25.0167847%	25.0076294%	49.9755859%	1.8311E-4	1.1930E8
10	25.0031471%	24.9999046%	49.9969482%	6.4850E-5	9.5113E8
11	25.0008702%	25.0006557%	49.9984741%	4.2915E-6	2.1719E11
12	25.0000600%	25.0000354%	49.9999046%	4.9174E-7	1.6542E13
13	25.0000064%	25.0000055%	49.9999881%	1.8626E-8	1.1529E16
$n = 24$					
14	25.0000003%	25.0000001%	49.9999996%	4.1910E-9	2.2774E17
15	25.0000001%	25.0000000%	49.9999999%	6.9849E-10	8.1986E18
16	25.0000000%	25.0000000%	50.0000000%	2.9104E-10	4.7224E19
17	25.0000000%	25.0000000%	50.0000000%	8.7311E-11	5.2471E20
$n = 32, 48, 64$					
14	25.0000003%	25.0000001%	49.9999996%	3.8417E-9	2.7103E17
15	25.0000000%	25.0000000%	49.9999999%	1.3097E-10	2.3320E20
16	25.0000000%	25.0000000%	50.0000000%	1.5461E-11	1.6733E22
17	25.0000000%	25.0000000%	50.0000000%	1.3642E-12	2.1492E24

Table 5: The components of the G_2 selection made with the linear approximation as a result of the ignored nonlinear contributions for SIMON with various number of rounds computed per cycle (d). The relative t-test value computed between G_2, G_0 is given which indicates how the signal scales down with increasing d. The corresponding relative data count scale, n, is also given as the number of measurements in each group required to maintain the $d = 1$ t-test result at larger d, provided that the standard deviation term in the t-test is unchanged and the event count is the same between the groups. This scale factor should multiply the number of events required for satisfactory key recovery at $d = 1$ to obtain an estimated event count for larger d. The case for G_0 is the same result with the index swap $2 \rightleftharpoons 0$.

d	Total Key Bits	Average Key Bits Recovered k_A	k_B	k_0	k_1	k_2	k_3	Total
1	32			32 (100.0%)				32 (100.0%)
2	64			31.36 (98.0%)	31.09 (97.2%)			62.45 (97.6%)
3	96	30.6 (95.7%)	30.1 (94.0%)	29.9 (93.5%)	28.2 (88.1%)	28.9 (90.3%)		87.0 (90.6%)
4	128	27.55 (86.1%)	22.36 (69.9%)	25.18 (78.7%)	22.45 (70.2%)	19.82 (61.9%)	19.45 (60.8%)	86.91 (67.9%)

Table 6: The average number of key bits recovered for SIMON n=32 on the Spartan 6 for the 11 encryption tests for $d = 1, 2, 3, 4$ where each dataset had 50k random plaintexts per cycle. The success of a recovery decreases with d due to the increasing contribution of the non linear terms. The success also decreases as additional key bits are recovered and errors in previously recovered bits propagate to the later keys. The recovery rate of k_A and k_B is shown only for d where the actual round keys are not directly obtained from k_A^* and k_B^* in the first pass. Note that a recovery of 50% is consistent with a random guess.

7.1 Pipelined SIMON

In a pipelined implementation, a given buffer is always overwritten by the same i^{th} round result. The power signal is then based on the result of two consecutive plaintext encryptions, requiring the attacker to know both plaintexts. For instance, if the first encryption is performed on $\mathbf{X}_A, \mathbf{Y}_A$ and the second is performed on $\mathbf{X}_B, \mathbf{Y}_B$, the signal on the buffers storing result for round $i+d$ is the Hamming distance which is $\propto \beta(\mathbf{Y}_{B,i+d} \rightarrow \mathbf{Y}_{A,i+d}) + \beta(\mathbf{X}_{B,i+d} \rightarrow \mathbf{X}_{A,i+d})$ where $\beta(x)$ is the function returning the number of bits set in x. The linear key term contributions cancel, leaving the attack to be performed only on the nonlinear terms. This reduces a source of propagating error in the nonlinear term attack but does not give the trivial result for $d = 1$. For both $d = 1, 2$, k_0 is recovered by a 2-bit AON attack on the register overwrite two rounds deep producing the signal

$$\begin{aligned} \beta(X^j_{B,i+2} \rightarrow X^j_{A,i+2}) &= X^{1+j}_{B,i+1} \cdot X^{8+j}_{B,i+1} + X^{1+j}_{A,i+1} \cdot X^{8+j}_{A,i+1} + C_{X,2}(\mathbf{X}_{A,i}, \mathbf{X}_{B,i})^j \\ &\propto (A^{1+j}_{B,i} + k^{1+j}_i) \cdot (A^{8+j}_{B,i} + k^{8+j}_i) + (A^{1+j}_{A,i} + k^{1+j}_i) \cdot (A^{8+j}_{A,i} + k^{8+j}_i) \end{aligned} \tag{19}$$

where the key independent values for the X and Y contributions for $d = 2$ are given by $C_{X,i+d}$ and $C_{Y,i+d}$. The key dependent values are the nonlinear terms of $\mathbf{X}$. To solve for k^{1+j}_i, only data is examined which has the property that $A^{8+j}_{A,i} = \overline{A^{8+j}_{B,i}}$ and $A^{1+j}_{A,i} = A^{1+j}_{B,i} \equiv A^{1+j}_{A/B,i}$ which reduces the signal to

$$\beta(X^j_{B,i+2} \rightarrow X^j_{A,i+2}) = (A^{1+j}_{A/B,i} + k^{1+j}_i) \cdot 1 + C_{X2}(\mathbf{X}_{A,i}, \mathbf{X}_{B,i})^j \tag{20}$$

which is solvable by the 2-bit AON method. The signal $\beta(X^j_{B,2} \rightarrow X^j_{A,2})$ yields k_0, The value of k_1 is found in the same way on $X^j_{B,3} \rightarrow X^j_{A,3}$ for $d = 1$ or $Y^j_{B,4} \rightarrow Y^j_{A,4}$ for $d = 1, 2$ using previously found round keys. The value of k_2 is then found from $X^j_{B,4} \rightarrow X^j_{A,4}$ and k_3 from $Y^j_{B,5} \rightarrow Y^j_{A,5}$ for $d = 1$ or from $X^j_{B,6} \rightarrow X^j_{A,6}$ from $d = 1, 2$.

The approach is the same for $d > 2$, although there will be some unknown nonlinear terms which are ignored in the linear approximation and which preferentially take on values of zero according to equation 16.

In general, the signal from $\beta(X^j_{B,i+d} \rightarrow X^j_{A,i+d})$ will contain $2(i + d - 1)$ nonlinear terms for the round keys $k_i, ..., k_{i+d-2}$ and the signal from $\beta(Y^j_{B,i+d} \rightarrow Y^j_{A,i+d})$ will contain $2(i+d-2)$ nonlinear terms for the round keys $k_i, ..., k_{i+d-3}$. The two lowest order nonlinear terms will be fixed in the first round key recovery, with each subsequent round key recovery fixing additional term pairs. To obtain an equivalent result as the standard implementation at $d = 5$ as seen in table 4, the number of unknown nonlinear terms must be at least four, resulting in a minimum depth of $d \geq 5$, where $\beta(Y^j_{B,i+5} \rightarrow Y^j_{A,i+5})$ will contain six nonlinear terms of which two will be fixed leaving four unknown terms. Compared to the PUR implementation, this approach requires a factor of four increase in the amount of data required from the fixing of the terms, and the number of nonlinear terms increases twice as fast as the coupled equation standard implementation, resulting in comparably stronger implementations for $d > 5$ over non-pipelined implementations.

8 Advanced Attacks on SIMON LUTs

It may be possible to mount a forward attack targeting individual LUTs in a SIMON implementation, where each LUT represents a single bit of a round function. When $d \geq 2$, the equation for a second round LUT is

$$\begin{aligned} X^j_{i+2} &= Y^j_{i+1} + k^j_{i+1} + X^{2+j}_{i+1} + X^{1+j}_{i+1} \cdot X^{8+j}_{i+1} \\ &= X^j_i + k^j_{i+1} + k^{2+j}_i + A^{2+j}_i \\ &\quad + (A^{1+j}_i + k^{1+j}_i) \cdot (A^{8+j}_i + k^{8+j}_i) \\ A^j_i &\equiv Y^j_i + X^{2+j}_i + (X^{1+j}_i \cdot X^{8+j}_i) \end{aligned} \tag{21}$$

Two equations are then constructed for the values entering the nonlinear AND term of the second round LUT.

$$\begin{aligned} X_{i+1}^{1+j} &= k_1^{1+j} + A_i^{1+j} \\ &= \kappa_A^{1+j} + A_i^{1+j} \\ X_{i+1}^{8+j} &= k_1^{8+j} + A_i^{8+j} \\ &= \kappa_A^{1+j} + \kappa_B^{1+j} + A_i^{8+j} \end{aligned} \tag{22}$$

If the plaintext is known, a hypothesis can be made for the key pair where the real key value is contained within the possible hypothesis $(k_i^{1+j}, k_i^{1+j} + k_i^{8+j}) \in (\kappa_A, \kappa_B)$.

Groups are then formed for the key hypothesis $(\kappa_A, \kappa_B) \in \{(0,0),(0,1),(1,0),(1,1)\}$ where the group G_2 represents that both input terms to the second level LUT are one, G_1 represents one input is one and the other is zero, and G_0 represents both being zero.

The t-test value is computed as was done for the register attack. The correct key hypothesis is determined as the one producing the largest $t_{2,0}^{(\kappa_A,\kappa_B)}$, although unlike in the register attack where a change in flip flop state from a previous state consumed more power than the unchanging flip flop, it cannot be determined if the largest t corresponds to G_2 or G_0. This is because the LUT will consume power depending on the function, input order (which inputs are connected to which address lines of the LUT), and relative time ordering of the input value changes (routing and placement effects). This behavior would differ from a similar attack on an ASIC implementation of SIMON where the power consumption of an AND gate would be dependent on its inputs alone and not correlated with the state of the other values of the round function terms. An unchanging input on a LUT could consume more power than the same input being changed depending on the other inputs and other input change order, or visa versa. Data used in this test was created with implementations using a constant fixed LUT definition for all round LUTs, eliminating uncertainty arising from signal input order. The ability to determine which ordering of G_2 or G_0 corresponds to the largest of $t_{2,0}^{\kappa_A,\kappa_B}, t_{0,2}^{\kappa_A,\kappa_B}$ and which corresponds to its compliment is a task requiring device characterization (device and implementation specific) which, with a known key, allows for the attack to determine which group (more or less power consumed) for which bit of each LUT of a given round corresponds to the correct key pairing. If device characterization is not possible, then 2^ν permutations per round must be examined which consider either G_2, G_0 for each bit, where ν is the bit width of the LUT tree. The necessary number of permutations in absence of device characterization is $2^{\mu\nu}$ where μ is the minimum of $d-1$ and m. The data needs to be reprocessed for each possible permutation to ensure proper data grouping, although this may still be a smaller effort than a brute force of the expected key strength of 2^{nm}.

If the correct k_1^{1+j} can be found from κ_A, then the known plaintext could be encrypted by a single round and the process could be repeated to find k_2 and so on up to k_{d-2}. The final k_{d-1} can be found with the $d = 1$ register attack.

8.1 Forward SIMON LUT Attack $d = 3$

With $d = 3$, the LUT attack can be performed to find k_i, k_{i+1}. The same dataset is used as for the $d = 3$ register attack. The number of correct recoveries of κ_A by bit position is presented in figure 4. The correct key recovery rate is insufficient for a practical attack.

The expected key recovery accuracy for $n = 32$ once the correct characterization permutation is used is

	1σ	2σ	3σ
k_i	26	22	7
k_{i+1}	17	11	4

where the number of bit positions are given that recover the correct key 68% of the time (1σ), 95% of the time (2σ), and 99.7% of the time (3σ). Note how the second round has worse performance, even though for this test the correct k_i was used to find the plaintext input to that round. This may be due to a temporal smearing effect from the placement and routing.

If the threshold is reduced, the number of bit positions that can recover a bit of k_i or k_{i+1} with a 68% confidence is 26 and 17 respectively. This attack therefore does not show the necessary reliability that was seen for the register type attack. While this type of attack may work for $\nu \ll n$, which is not tested here, the case of $\nu = n$ for $n = 32$ shows significant variation in the power consumption of the LUT which is dependent not only on the input value of the nonlinear term components but on the values of the other inputs and the timing order in which the inputs change value. The plot of the t-test value shown in figure 5 illustrates the degenerate signal which has a substantially smaller separation in the second round. Device characterization with a known key attack could be used to determine the G_2, G_0 ordering per individual LUT for layout specific effects.

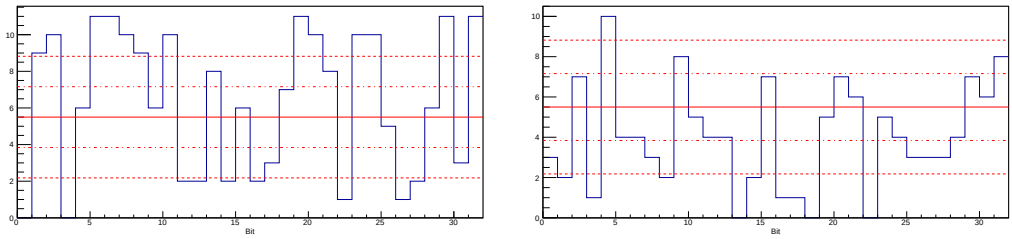


Fig. 4: For each bit position (x-axis) examined for the $d = 3$ forward SIMON $n = 32$ LUT attack the number of κ_A correct bits for the 11 cycle test is presented (y-axis) for the attack on the LUTs for k_i (left) and k_{i+1} (right). A bit position with a large number of bits correct has the expected group ordering $G_2 > G_0$ while a bit position with a small number of bits correct has the inverted group ordering $G_0 > G_2$. Those with an average near 5.5 (solid red line) do not exhibit the ability to distinguish the correct κ_A. The dashed lines indicate $1\sigma, 2\sigma$ from the average case of a random guess for κ_A.

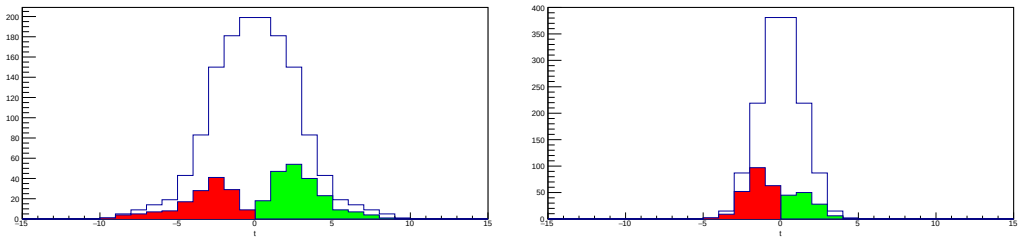


Fig. 5: The t values (x-axis) over the the 11 cycle test for the $d = 3$ forward SIMON $n = 32$ LUT attack carried out over each bit is presented for k_i (left) and k_{i+1} (right). The t corresponding to the expected group ordering $G_2 > G_0$ for the correctly identified κ_A is shown in green while the t corresponding to the inverted group ordering $G_0 > G_2$ for the correctly identified κ_A is shown in red. The statistical separation between the correct t (from G_2, G_0 or G_0, G_2) from the degenerate case t (from G_1, G_1) is evident for the k_i recovery but is substantially reduced for the k_{i+1} recovery due to a wider variance in the timing of the changing signals in the second round LUTs compared to the first round LUTs.

8.2 Reverse SIMON LUT Attack $d = 3$

A reverse attack can also be carried out on the SIMON LUTs using the encryption result, X_T, Y_T. A hypothesis for k^1_{T-1}, k^8_{T-1} fixes the input to the second to last round AND.

$$\begin{aligned} X^{1+j}_{T-2} &= k^{1+j}_{T-1} + Y^{1+j}_T + X^{3+j}_T + X^{2+j}_T \cdot X^{9+j}_T \\ X^{8+j}_{T-2} &= k^{8+j}_{T-1} + Y^{8+j}_T + X^{10+j}_T + X^{9+j}_T \cdot X^{17+j}_T \end{aligned} \tag{23}$$

If the correct key hypothesis is chosen, the two inputs will be correctly grouped as both zero or both one. In the case of both zero, this AND as well as two others will have fixed output results of zero. In the case of both one, this AND will have a fixed result of one with two others taking equal probable outputs. If one is zero and the other is one, this AND will have fixed output of zero and one other AND will have a fixed output of zero. This again leads to two nearly degenerate G_1, a G_2, and a G_0 grouping. Again, it is not known if fixing a particular AND output will contribute more or less power to the operation of the LUT, so either device characterization or the necessary permutations must be carried out. If k_{T-1} can be recovered, the process can be repeated to recover k_{T-2} and so on. It is seen that this reverse attack does not perform as well as the forward attack, with only a few bits able to be recovered consistently with the expected key recovery accuracy for the $d = 3$ reverse attack on the SIMON $n = 32$ LUTs once the correct characterization permutation is used, where the number of bit positions are given that recover the correct key 68% of the time (1σ), 95% of the time (2σ), and 99.7% of the time (3σ).

	1σ	2σ	3σ
k_{i+2}	6	2	1
k_{i+1}	7	4	1

The splitting of t for the correct group is not as clear, with most bit positions unable to perform a statistically significant identification.

9 Advanced Attacks on SPECK $d \geq 2$

When attacking SPECK with $d \geq 2$, there are key-dependent carry terms from the addition which are uniformly distributed for random plaintext. For this reason, it is necessary to fix the value of these carry terms. The equations for $\mathbf{X}, \mathbf{Y}$ are

$$\begin{aligned} X^j_{i+1} &= \kappa_A + \kappa_C + X^{\alpha+j}_i + Y^j_i + C^j_i \\ Y^j_{i+1} &= \kappa_A + \kappa_B + X^j_{i+1} + Y^{\beta+j}_i \end{aligned} \tag{24}$$

where κ_A are linear key terms in common between $\mathbf{X}$ and $\mathbf{Y}$, κ_B, κ_C are linear key terms unique to $\mathbf{Y}$ and $\mathbf{X}$ respectively, as given in table 7, and α, β are the parameters of the particular SPECK implementation.

A general approach allows for the filtering of the random plaintext to fix the carry terms

$$\begin{aligned} C^{j+1}_i &= (W^{\alpha+j}_i + k^{\alpha+j}_i) \cdot (Z^j_i + k^j_i) + (W^{\alpha+j}_i + k^{\alpha+j}_i + Z^j_i + k^j_i) \cdot C^j_i \\ C^0_i &= 0 \end{aligned} \tag{25}$$

The carry term C^{j+1}_i can be fixed by examining all possible key bit values in the carry terms of the expansion and for each combination assigning a subset of data which results in fixed carry terms. By selecting data which has the property $W^{\alpha+j}_i = \kappa^{\alpha+j}_i$ and $Z^j_i = \kappa^j_i$, the resulting dataset has the property that for every equation examined the carry terms will be fixed at zero provided that the κ are correct. If the κ are partially incorrect, then the data will be misclassified reducing the maximum t value towards zero from a random contribution through the exposed C^j_i. In the case where both key bits are incorrect so that

$W_i^{\alpha+j} = \overline{\kappa_i^{\alpha+j}}$ and $Z_i^j = \overline{\kappa_i^j}$ the carry bit will be fixed at one. Since all of the carry bit contributions appear in a linear combination, the overall value of a set of fixed carry terms is either zero or one. As the κ contained within a single carry term may be flipped changing the fixed state of that term, the ability to resolve the correct case from the degenerate case depends on the term content of $\kappa_A, \kappa_B, \kappa_C$. While the correct and complimentary carry bits produce degenerate groups, the relative values do provide useful information as to whether the two key bits in a given multiplication term take on the same or different values from each other. As a result, once a few key bits have been recovered it is possible to extend that information to recover many more key bits without any further data processing. Once the carry contributions are fixed, the exposed key terms may cancel the normal linear key terms, thus producing a reduced set of free key terms.

d	κ_A	κ_B	κ_C
1	(1) k_0	(0)	(0)
2	(3) k_0, k_0^{α}, k_1	(1) k_0^{β}	(0)
3	(5) k_0, $k_0^{2\alpha}$, k_1, k_1^{α}, k_2	(3) $k_0^{\alpha+\beta}$, $k_0^{2\beta}$, k_1^{β}	(1) k_0^{β}
4^a	(10) k_0, k_0^{α}, $k_0^{2\alpha}$, $k_0^{3\alpha}$, $k_0^{2\beta}$, k_1, $k_1^{2\alpha}$, k_2, k_2^{α}, k_3	(7) $k_0^{\alpha+2\beta}$, $k_0^{2\alpha+\beta}$, $k_0^{3\beta}$, k_0^{β}, $k_1^{\alpha+\beta}$, $k_1^{2\beta}$, k_2^{β}	(1) k_1^{β}
4^b	(10) k_0, k_0^{α}, $k_0^{2\alpha}$, $k_0^{3\alpha}$, $k_0^{2\beta}$, k_1, $k_1^{2\alpha}$, k_2, k_2^{α}, k_3	(5) $k_0^{2\alpha+\beta}$, k_0^{β}, $k_1^{\alpha+\beta}$, $k_1^{2\beta}$, k_2^{β}	(1) k_1^{β}

Table 7: The linear term key components of SPECK ($n = 32, 48, 64$) for various d where $\mathbf{X} \propto \boldsymbol{\kappa}_A + \boldsymbol{\kappa}_C$ and $\mathbf{Y} \propto \boldsymbol{\kappa}_A + \boldsymbol{\kappa}_B$. The fixing of the carry contributions cancels some of the standard linear key terms for $d \geq 4$, here shown as the linear terms ($d = 4^a$) and resulting terms after cancelation ($d = 4^b$).

There is a concept of carry term chaining, where a key term that appears in one carry term may also appear in one or more additional carry terms. The compliment of the key hypothesis of one carry term requires both of the key hypothesis in all chained carry terms to also invert to ensure that these carry terms remain constant. This results in a reduced number of degenerate groupings which are indistinguishable from the correct solution. A list of the carry terms, carry chains, and number of degenerate correct solution states is given in table 8 for $n = 32, 48, 64$. Note that for $n = 24$ some of the terms cancel due to index wrapping.

For a given carry term in a carry chain, C_i^j there is a key hypothesis for the relationship between the chained key terms $\kappa_{i-1}^{\alpha+j-1}, \kappa_{i-1}^{j-1}$ as obtained from equation 25. Independent of whether this is the correct key hypothesis or its conjugate, one of the conditions of $k_{i-1}^{\alpha+j-1} = k_{i-1}^{j-1}$ or $k_{i-1}^{\alpha+j-1} = \overline{k_{i-1}^{j-1}}$ will hold. With a single carry chain containing at least $\lceil \frac{n}{|\alpha|} \rceil$ key terms spaced every $|\alpha|$ and thus at least $\lceil \frac{n}{|\alpha|} \rceil - 1$ carry terms, it is only necessary to permute a single key value in the carry chain to obtain all $\lceil \frac{n}{|\alpha|} \rceil$ key terms. If there are $|\alpha|$ adjacent chains, only α bits need to be permuted to recover an entire round key. A d implementation recovers $d-2$ round keys from carry chains and the remaining 2 round keys from the $d = 2$ method. As seen in table 8, for $d \geq 3$ at least two chains of two terms are contained at the $d-2$ level spaced one index apart, with additional and longer chains given for earlier rounds reducing the number of required equation pairs to be examined by half.

d	G_2	G	Carry Chain	X	Y	N_X	N_Y
2	2^1	2^2	C_1	C_1	C_1	1	1
3	2^4	2^{10}	C_1, C_1^{α}	C_1, C_1^{α}	C_1, C_1^{α}	2	2
			C_1^{β}		C_1^{β}	0	1
			C_2	C_2	C_2	1	1
			$C_1^{\alpha-1}$, C_1^{-1}			0	0
4	2^{11}	2^{30}	C_1	C_1	C_1	1	1
			$C_1^{\alpha+\beta}$, C_1^{β}	C_1^{β}	$C_1^{\alpha+\beta}$	1	1
			$C_1^{2\alpha}$	$C_1^{2\alpha}$	$C_1^{2\alpha}$	1	1
			$C_1^{2\beta}$		$C_1^{2\beta}$	0	1
			C_2, C_2^{α}	C_2, C_2^{α}	C_2, C_2^{α}	2	2
			C_2^{β}		C_2^{β}	0	1
			C_3	C_3	C_3	1	1
			$C_1^{\alpha-1}$, $C_1^{2\alpha-1}$, C_1^{-1}			0	0
			$C_1^{\alpha+\beta-1}$, $C_1^{\beta-1}$			0	0
			$C_2^{\alpha-1}$, C_2^{-1}			0	0
			$C_1^{\alpha-2}$, $C_1^{2\alpha-2}$, C_1^{-2}			0	0
5	2^{19}	2^{60}	C_1, C_1^{α}, $C_1^{2\alpha}$, $C_1^{3\alpha}$	C_1, C_1^{α}, $C_1^{2\alpha}$, $C_1^{3\alpha}$	C_1, C_1^{α}, $C_1^{2\alpha}$, $C_1^{3\alpha}$	4	4
			$C_1^{\alpha+2\beta}$, $C_1^{2\beta}$	$C_1^{2\beta}$	$C_1^{\alpha+2\beta}$, $C_1^{2\beta}$	1	2
			$C_1^{\alpha-3}$, $C_1^{2\alpha+\beta}$, $C_1^{3\alpha-3}$, C_1^{β}		$C_1^{2\alpha+\beta}$, C_1^{β}	0	2
			$C_1^{\alpha-1}$, $C_1^{2\alpha-1}$, $C_1^{3\beta}$, C_1^{-1}		$C_1^{3\beta}$	0	1
			C_2	C_2	C_2	1	1
			$C_2^{\alpha+\beta}$, C_2^{β}	C_2^{β}	$C_2^{\alpha+\beta}$	1	1
			$C_2^{2\alpha}$	$C_2^{2\alpha}$	$C_2^{2\alpha}$	1	1
			$C_2^{2\beta}$		$C_2^{2\beta}$	0	1
			C_3, C_3^{α}	C_3, C_3^{α}	C_3, C_3^{α}	2	2
			C_3^{β}		C_3^{β}	0	1
			C_4	C_4	C_4	1	1
			$C_1^{\alpha+\beta-1}$, $C_1^{2\alpha+\beta-1}$, $C_1^{\beta-1}$			0	0
			$C_1^{\alpha+2\beta-1}$, $C_1^{2\beta-1}$			0	0
			$C_2^{\alpha-1}$, $C_2^{2\alpha-1}$, C_2^{-1}			0	0
			$C_2^{\alpha+\beta-1}$, $C_2^{\beta-1}$			0	0
			$C_3^{\alpha-1}$, C_3^{-1}			0	0
			$C_1^{\alpha-2}$, $C_1^{2\alpha-2}$, $C_1^{3\alpha-2}$, C_1^{-2}			0	0
			$C_1^{\alpha+\beta-2}$, $C_1^{2\alpha+\beta-2}$, $C_1^{\beta-2}$			0	0
			$C_2^{\alpha-2}$, $C_2^{2\alpha-2}$, C_2^{-2}			0	0

Table 8: For the first few d of SPECK for $n = 32$, the number of degenerate G_2 groups formed compared to the total number of groups (G) when the terms in the carry chains are fixed is presented. Terms which appear with an even count in X or Y (N_X, N_Y) have no effect when the chain value is flipped, while those with an odd count in X or Y propagate a flip to that equation. Terms with zero count in X and Y are inner terms of higher level carries which appear with even contribution in the parent carry and do not effect the key hypothesis in the parent carry when flipped. Similar terms appear for $d < 5$ for $n = 48, 64$ with the $d = 5$ having $G = 2^{65}$.

The minimum attack count, A_C, required to obtain carry chains which define all round key bits for rounds $0, ..., i_c - 1$ with $i_c = \min(d-2, m)$ is then given as

$$A_C = \frac{|\alpha|}{2}\left\lceil \frac{n-|\alpha|}{b|\alpha|} \right\rceil \tag{26}$$

where $2n$ is the block size, $|\alpha|$ is the magnitude of the SPECK α shift parameter, and b is the maximum number of chain terms in the longest two chains offset by an index of one at the i_c level, which is also the number of additional key bits obtained from the chain relationship knowing one of the key values in the chain. The $\frac{|\alpha|}{2}$ scale factor accounts for the two equal length carry chains offset by a single index per attack. The number of brute force permutations of key bits at $d-2$ is $2^{|\alpha|}$ and is approximated for the i_c rounds as $2^{i_c|\alpha|}$, although it may be less as chains corresponding to $i < d-2$ have more than two chains which are also longer than three carry terms each and may provide additional relationships between round key bit values at those levels. Additional terms may be calculated at more frequent intervals providing a means of cross checking and error detection. A closed chain may be formed which gives relationship values for a given key bit with at least two other key bits, including the key bits that are brute forced thus providing an additional means of error detection.

If the number of all terms appearing in both $\mathbf{X}$ and $\mathbf{Y}$ from a particular carry chain is either even or odd then κ_A cannot be determined, but κ_B can be determined in the case of an empty κ_C, a condition which applies to $d = 2$ only. For $d \geq 3$, there are combinations of both even, both odd, or one even and one odd term count in $\mathbf{X}/\mathbf{Y}$ for a given carry chain. This results in the inability to determine the relationship between $\kappa_A, \kappa_B, \kappa_C$. In this case, all of the useful information is in the carry terms.

The number of degenerate solutions indistinguishable from the correct solution along with the total number of groups that are formed with all possible carry key hypothesis are presented in table 9. For all n, the total group size begins to become unmanageable for $d \geq 4$, requiring 2^{27} groups for $n = 24$ and 2^{30} groups for $n = 32, 48, 64$. Even if an attack were to by chance find a group degenerate with G_c, for $d \geq 4$ the minimum number of groups that would need to be examined assuming evenly distributed groups degenerate with G_c would still not be feasible with 2^{17} groups for $n = 24$, and 2^{19} groups for $n = 32, 48, 64$. Grouping can be performed on the key relationships rather than the key values, resulting in a total of $m_{G_c} = \frac{G_t}{G_c}$ groups which must be examined. For such an attack, a sufficient number of events for each group being examined would need to be generated for each bit under attack. While an attacker capable of constructing plaintext could potentially carry out this attack, a random plaintext sample would not be expected to have samples which preferentially fell into some groups more often than others.

	$n=24$			$n=32$			$n=48$			$n=64$		
d	G_c	G_t	m_{G_c}	G_c	G_t	m_{G_c}	G_c	G_t	m_{G_c}	G_c	G_t	m_{G_c}
2	2^1	2^2	2^1	2^1	2^2	2^1	2^1	2^2	2^1	2^1	2^2	2^1
3	2^4	2^{10}	2^6	2^4	2^{10}	2^6	2^4	2^{10}	2^6	2^4	2^{10}	2^6
4	2^{10}	2^{27}	2^{17}	2^{11}	2^{30}	2^{19}	2^{11}	2^{30}	2^{19}	2^{11}	2^{30}	2^{19}
5	2^{18}	2^{51}	2^{33}	2^{19}	2^{60}	2^{41}	2^{19}	2^{65}	2^{46}	2^{19}	2^{65}	2^{46}
6	2^{26}	2^{75}	2^{49}	2^{27}	2^{92}	2^{65}	2^{27}	2^{108}	2^{81}	2^{28}	2^{112}	2^{84}

Table 9: The number of degenerate groups indistinguishable from the correct key set including the correct case (G_c), the total number of groups (G_t), and the expected minimum number of groups to be examined before a group in the degenerate correct group set is observed ($m_{G_c} = G_t/G_c$) assuming an even distribution of the degenerate groups for the first few d of SPECK.

For a given key hypothesis, a number of bits of the input plaintext all take the same fixed value as they are included in the carry terms. The other bits which enter into the equation are free terms which vary from the random plaintext. Any other bit of the plaintext is unused in an attack on a single bit. For this reason, an attacker may carefully construct a sample of random plaintext which only includes bit variations on the free and fixed terms for $|\alpha|$ contiguous bits.

The special case of $d = 2$ requires a single pass to group the data into 16 groups (κ_A, κ_B and both key bits of the single carry term), recovering k_0 from κ_B when data is grouped to fix the single carry term. The found k_0 is then used with the already recovered κ_A to recover k_1 after determining which carry chain group is correct by examining the actual corresponding k_0 relationship. No brute force step is required.

In the cases of $d \geq 3$ where $\kappa_A, \kappa_B, \kappa_C$ all have non fixed terms, the chained keys in the carry terms can be used with a brute force attack on a fraction of bits to obtain key candidates which must then be tested. For the specific case of $d = 3, m = 4$ a second cycle must be examined with each key hypothesis for k_0, k_1, k_2. The $d = 2$ attack method can be carried out on each key hypothesis for $d = 3, m = 2, 3, 4$ with κ_0, for $d = 4, m = 3, 4$ with κ_0, κ_1, and for $d = 4, m = 4$ with $\kappa_0, \kappa_1, \kappa_2$. and which provide C_1, C_2 for $d = 3$ and C_1, C_2, C_3 for $d = 4$ requiring $2^{2|\alpha|}$ and $2^{3|\alpha|}$ guesses respectively to recover complete probable k_0^*, k_1^* and k_0^*, k_1^*, k_2^* keys. However, once the $d - 2$ keys are found at a cost of $2^{(d-2)|\alpha|}$, the remaining 2 keys can be found with another pass using the $d = 2$ method. Once the keys $k_{i+0}, ...k_{i+d-3}$ have been set via brute force and carry chain key relationships, the equations for $\mathbf{X}, \mathbf{Y}$ take the form of coupled equations with inverted hierarchy

$$\begin{aligned} X_{i+d}^j &= \kappa_A^j + \kappa_B^j + C_{i+d-1}^j + f(\mathbf{X}_i, \mathbf{Y}_i)^j \\ Y_{i+d}^j &= \kappa_A^j + C_{i+d-1}^j + g(\mathbf{X}_i, \mathbf{Y}_i)^j \end{aligned} \tag{27}$$

where the functions f, g include the plaintext component. The solution for the remaining k_{i+d-2}, k_{i+d-1} follows the $d = 2$ solution. In the case of $d = 3, m = 4$ the attack also needs to also be performed on the second cycle.

The effective key strength for various configuration is presented in table 10. The case of $d = 3$ is found to require a manageable data size of a minimum of 64 groups and a maximum of 1,000 groups to be examined, each of which must contain a number of random plaintexts to provide statistically significant samples. For $d \geq 4$ although the number of permutations required is small compared to the expected key complexity, the number of random data samples required is large enough to make the attack unfeasible. With $d = 4$, a minimum of 131k (2^{17}) groups must be examined to find a group degenerate with the correct group. A total of 130 million groups are formed for $n = 24$ and over one billion groups for larger n. As each group must contain a number of random plaintexts to provide statistically significant samples, the carefully formed minimum group set with each group containing at least ten entries would be comparable to the data set size required for a comparable SIMON $d = 5$ attack complexity. That limited number of entries would probably be insufficient for a statistically significant measurement.

d	m	Chain Attack Count	Keys From Chains	Expected Key Strength	Minimum Effective Strength	Needs $d=2$ Pass	Needs Second Cycle	SPECK 64 $\|\alpha\|=8, n=32$		SPECK 128 $\|\alpha\|=8, n=64$	
								Attack	MES	Attack	MES
3	2	$\frac{\|\alpha\|}{2}\lceil(\frac{n-\|\alpha\|}{2\|\alpha\|})\rceil$	1	2^{2n}	$2^{\|\alpha\|}$	x	-	8	2^8	16	2^8
	3	$\frac{\|\alpha\|}{2}\lceil(\frac{n-\|\alpha\|}{2\|\alpha\|})\rceil$	1	2^{3n}	$2^{\|\alpha\|}$	x	-	8	2^8	16	2^8
	4	$\|\alpha\|\lceil(\frac{n-\|\alpha\|}{2\|\alpha\|})\rceil$	2	2^{4n}	$2^{2\|\alpha\|}$	x	x	16	2^{16}	32	2^{16}
4	2	$\frac{\|\alpha\|}{2}\lceil(\frac{n-\|\alpha\|}{2\|\alpha\|})\rceil$	2	2^{2n}	$2^{2\|\alpha\|}$	-	-	8	2^{16}	16	2^{16}
	3	$\frac{\|\alpha\|}{2}\lceil(\frac{n-\|\alpha\|}{2\|\alpha\|})\rceil$	2	2^{3n}	$2^{2\|\alpha\|}$	x	-	8	2^{16}	16	2^{16}
	4	$\frac{\|\alpha\|}{2}\lceil(\frac{n-\|\alpha\|}{2\|\alpha\|})\rceil$	2	2^{4n}	$2^{2\|\alpha\|}$	x	-	8	2^{16}	16	2^{16}
5	2	$\frac{\|\alpha\|}{2}\lceil(\frac{n-\|\alpha\|}{3\|\alpha\|})\rceil$	2	2^{2n}	$2^{2\|\alpha\|}$	-	-	4	2^{16}	12	2^{16}
	3	$\frac{\|\alpha\|}{2}\lceil(\frac{n-\|\alpha\|}{2\|\alpha\|})\rceil$	3	2^{3n}	$2^{3\|\alpha\|}$	-	-	8	2^{24}	16	2^{24}
	4	$\frac{\|\alpha\|}{2}\lceil(\frac{n-\|\alpha\|}{2\|\alpha\|})\rceil$	3	2^{4n}	$2^{3\|\alpha\|}$	x	-	8	2^{24}	16	2^{24}
6	2	$\frac{\|\alpha\|}{2}\lceil(\frac{n-\|\alpha\|}{4\|\alpha\|})\rceil$	2	2^{2n}	$2^{2\|\alpha\|}$	-	-	4	2^{16}	8	2^{16}
	3	$\frac{\|\alpha\|}{2}\lceil(\frac{n-\|\alpha\|}{3\|\alpha\|})\rceil$	3	2^{3n}	$2^{3\|\alpha\|}$	-	-	4	2^{24}	12	2^{24}
	4	$\frac{\|\alpha\|}{2}\lceil(\frac{n-\|\alpha\|}{2\|\alpha\|})\rceil$	4	2^{4n}	$2^{4\|\alpha\|}$	-	-	8	2^{32}	16	2^{32}

Table 10: Details of the advanced attack on SPECK are shown for the number of rounds computed per cycle, d, and SPECK parameters of m, block size $2n$, and the magnitude of the shift parameter, α. The chain attack count indicates the number of index positions at which the DGPA $\delta = 2$ attack is carried out by fixing carry terms based on plaintext values in order to find the correct G_C or equivalent degenerate groups required to describe the bit relationships between the specified number of round keys which are obtained from these carry chains. The expected key strength is 2^{mn}. For each round key recovered using a chain which only defines relationships between key bits spaced at a distance of $|\alpha|$, a brute force effort is required. Permuting the initial $|\alpha|$ bits of each round key results in a minimum effective strength (MES) which is substantially reduced from the expected key strength. When the m round keys are not all recovered in the $d-2$ chains, a $d=2$ type pass is used to recover two more round keys. Implementations resulting in $m+1$ keys recovered after the $d=2$ pass can use the extra key as a lower cost verification of full key recovery with the key scheduler. A second cycle is required for the $d=3, m=4$ resulting in a doubling of the chain attack count and bits that must be brute forced, although the $d=2$ pass is not required in the second cycle. The recovered key is then checked against a known plaintext/ciphertext pair to determine if it is correct. More complicated attacks are possible which perform more than the stated minimum chain attack count and which use additional chains to verify the found chain relationships prior to the brute force step. Example chain attack counts and the minimum effective strength are presented for variants of SPECK 64 and SPECK 128.

9.1 Pipelined SPECK

Similar to pipelined SIMON, linear key terms cancel resulting in key terms only appearing in the nonlinear terms. For SPECK $d = 1$ the signal must be observed at a spacing of at least two rounds. For all d, all carry terms must still be fixed since they appear in duplicate in the form of

$$\begin{aligned} C_{A,i}^{j+1} + C_{B,i}^{j+1} = (W_{A,i}^{\alpha+j} + k_i^{\alpha+j}) \cdot (Z_{A,i}^{j} + k_i^{j}) + (W_{A,i}^{\alpha+j} + k_i^{\alpha+j} + Z_{A,i}^{j} + k_i^{j}) \cdot C_{A,i}^{j} \\ + (W_{B,i}^{\alpha+j} + k_i^{\alpha+j}) \cdot (Z_{B,i}^{j} + k_i^{j}) + (W_{B,i}^{\alpha+j} + k_i^{\alpha+j} + Z_{B,i}^{j} + k_i^{j}) \cdot C_{B,i}^{j} \end{aligned} \tag{28}$$

and they are only fixed when $W_A^{\alpha+j} + Z_A^j = W_B^{\alpha+j} + Z_B^j = k_i^{\alpha+j} + k_i^j$, thus increasing the number of random plaintexts required for an attack by a factor of 2.

The remaining terms will have a non zero result for half of the filtered random plaintexts when $W_{A,i}^{\alpha+j} = Z_{B,i}^j = k_i^{\alpha+j}, W_{B,i}^{\alpha+j} = Z_{A,i}^j = k_i^j$ or $W_{A,i}^{\alpha+j} = Z_{B,i}^j = k_i^j, W_{B,i}^{\alpha+j} = Z_{A,i}^j = k_i^{\alpha+j}$, resulting in

$$C_{A,i}^{j+1} + C_{B,i}^{j+1} = k_i^{\alpha+j} + k_i^j \tag{29}$$

In this approach, fixing the carry terms serves to increase the number of plaintexts required by a factor of 2× for every carry term. The exposed key terms then enter the linear equation and solving for them is trivial using the methods previously described. For $d \geq 3$ key relationships are exposed as $\kappa_A, \kappa_B, \kappa_C$, and for $d = 1, 2$ keys are solved for directly. As a result, the minimum number of groups from table 9 should be multiplied by the exponent from the G value (corresponding to the number of variables fixed in the carry terms) from table 8 to account for the increase in plaintexts required when fixing the carry contributions. This results in a minimum depth requirement still of $d \geq 4$ for pipeline SPECK, although for greater d, the protection is increasingly enhanced.

10 Rekeying Guidance

The periodic updating of an encryption key of a field deployed device can have a side channel mitigation effect by reducing the sample size of encryptions performed with the same key.

Short of a device receiving a new key from an external source, the device would need to create a new key on its own. The SIMON and SPECK key schedule algorithms currently have no means to do this and the definitions of each would need to be extended to accommodate self key updating. The updated key must be saved as a starting point upon power cycling or must otherwise be recoverable from the original key. There may be a power cycling vulnerability associated with key updating if upon power cycling the key state obtained from a key rotation key scheduler is lost, so any on device key rotation must save state to nonvolatile memory which may create another opportunity for attack.

A natural starting place for key updating would be to run the key scheduler beyond the T rounds to obtain the next m round keys which could be used as the next key. However, a complete cryptanalysis of this approach would need to be carried out and this is beyond the scope of this project. It has been observed through rekeying tests of many randomly generated keys that such a direct approach repeats keys after a relatively short number of rekeyings. Depending on the particular key, a rekeying of this form may increase the number of data points required to be captured by as much as a factor of one thousand. The key might be updated after a set number of encryptions or might be updated whenever a particular state is seen (such as three particular bits in the encrypted X are all ones). An attack would not be able to gather the necessary number of encryptions prior to the key change, and would either need to extend the attack to account for different keys obtained in this manner, or would need to wait until the original key was repeated. It may be useful to reserve a byte or more of plaintext to contain the number of updates that have occurred.

This way a decrypting system could iterate through updated keys until a key can successfully decrypt a cipher text and recover the correct key update count.

Some general guidelines regarding rekeying are presented taking into account the results of this work. Rekeying should not be performed by transmitting an unencrypted key over an I/O pin from another device even if the data wire is unexposed because the key power signal can be read directly as was seen on the Spartan V_{IO} supply. The rekeying process needs to be encrypted. A possible FPGA solution is a complete or partial reconfiguration through an encrypted bitstream which takes advantage of the dedicated Spartan AES configuration protection. The new key can also be encrypted using the old SIMON/SPECK key with the disadvantage that there is no forward security but the advantage that if the key is changed frequently enough then an attacker would not be able to acquire the necessary quantity of data to carry out a simple attack. If the key expansion is not protected with a threshold implementation then the average power in a computational cycle involving key information will change upon a rekeying indicating that a rekeying occurred.

The rate of rekeying will be dependent on the algorithm, device, and desired equivalent strength of the implementation. As an example, a minimum rekeying rate would be that necessary to limit the recovery of key bits to a portion of the total key. The recovery rate of the correct groups from DGPA $\delta = 2$ equation pairs used in each step or iteration is examined for SIMON and SPECK, where a recovery rate of $\frac{3}{4}n$, $n-1$, and n of the groups, designated $R^{\frac{3}{4}n}$, R^{n-1}, and R^n, respectively, are used as reference points.

The bit locations of the incorrectly recovered groups or associated key bits are expected to correspond to lower t values than correctly recovered groups when dealing with high statistic samples. At lower statistic samples, random fluctuations may place the t value of an incorrect group higher than a correct group. As inferred from the MSP SPECK 64/128 $d = 1$ first round key recovery example of figure 3, the largest incorrect t value should be at or below the average t value of the correctly recovered bits near $R^{\frac{3}{4}n} \approx 800$ plaintexts and less than the t value of all correctly recovered bits at ≈ 1200 plaintexts which is far below $R^{n-1} \approx 4000$ plaintexts. For this discussion, the assumption is made that incorrectly recovered groups will in the average case have a t value in the lower half of all recovered groups at $R^{\frac{3}{4}n}$ and will have a t value in the lower quarter of all recovered groups at R^{n-1}.

In the best case, the t ordering will be correct and the number of incorrectly recovered equation pairs will be exactly what is expected based on the number of random plaintext examined. This results in a correct key recovered for the first possible scenario examined for $d = 1$ cases, and simply requires a permutation over the other possible group values for the identified incorrect locations for $d \geq 2$. The worst case scenario assumes that the incorrectly recovered groups all have t values greater than all correct groups and that the recovery checks this condition only after exhaustively checking all other possibilities. Both the worst and average cases assume that the number of incorrect groups may not be exactly what is expected based on the number of plaintext examined.

The number of choices, $C_{R^{n_w}}$, of the average number of incorrect group locations, n_w, for a block size of $2n$ at R^{n-n_w}, assuming all incorrect groupings fall in the lowest n_t of the n total t values is

$$C_{R^{n-n_w}} = \binom{n_t}{n_w} + \sum_{i=1}^{\Delta} \left[\binom{n_t}{n_w + i} + \binom{n_t}{n_w - i} \right] \tag{30}$$

where Δ is maximum deviation of the the number of incorrect groups from the expected average which is calculated at a multiple of the standard deviation, $\sigma_{R^{n-n_w}}$ at R^{n-n_w} for a desired confidence level. Terms in the sum are only included for which $n_w + i \leq n_t$ and $n_w - i \geq 0$. The average case, $C^A_{R^{n_w}}$ assumes $n_t = \frac{n}{2}$ for $R^{\frac{3}{4}n}$ and $n_t = \frac{n}{4}$ for R^{n-1}. The worst case, $C^W_{R^{n_w}}$, assumes $n_t = n$ for all n_w. From the Spartan 64/128 $d = 1$ key recovery rate data used to obtained the fit as shown in table 3, it was also observed that $\sigma_{R^{\frac{3}{4}n}} \approx 3, \sigma_{R^{n-1}} \approx 1$ bits which is used as an assumption throughout these calculations.

An estimation of the effective key strength taking into account the variance in the expected number and location of incorrect group recoveries for the event counts corresponding to $R^{\frac{3}{4}n}, R^{n-1}, R^n$ is given in table 11 for SIMON and table 12 for SPECK. SIMON results use the number of events from the AON X test for $d = 1$ for $R^{\frac{3}{4}n}, R^{n-1}, R^n$. SPECK results use the number of events from the G_1 determined test for $d = 1$ for $R^{\frac{3}{4}n}$ and R^{n-1} assuming a minimum of 10 events per group to obtain $R^{\frac{3}{4}n}$ which is expected to actually yield slightly more than $\frac{3}{4}n$ correct bits (25 bits for n=32, 52 bits for n=64). The R^n events come from the AON Y $d = 1$ test. These assumptions are conservative estimates for $d \geq 2$ which are expected to require more events than simply scaling the $d = 1$ results as the attack methods should also partially follow the $d = 2$ results for some steps. Furthermore, the results obtained from table 3 correspond to the correct number of bits recovered, not the correct number of groups. Using these values as an approximation results in an overestimation of the G_2 component and thus an underestimation of the effort required.

For both SIMON and SPECK, l loops are performed with $l \equiv \lceil \frac{m}{d} \rceil$. Each loop examines a different power measurement and provides up to d extracted keys $k_i, ..., k_{i+d-1}$ with $i = 0, 1d, 2d, ..., (l-1)d$. Following the one or more DGPA passes and resulting key recoveries of a given loop, all possible incorrectly recovered groups must be examined with subsequent loops and corresponding additional DGPA passes being performed for each permutation. An alternate approach would be to perform the DGPA attack once, and then to perform the permutations accounting for propagation of error. While this approach is less computationally intensive per permutation, the number of permutations is greater as the likelihood of a failed group recovery increases due to the inclusion of incorrectly recovered keys from previous rounds with the exception of SPECK which has a constant effect for $d \geq 6$. Once a key hypothesis is found it must be tested against a known plaintext/ciphertext. However, for some instances, a $m + 1$ key is also found at no additional cost from a $d = 2$ type pass, allowing for a single cycle of the key expansion algorithm to be used to filter out incorrectly recovered keys more quickly.

Within a SIMON loop, single passes are performed for $d = 1, 2$ where the keys are extracted from the κ values in the linear terms. The best case for all d assumes that the first ordering examined places exactly the expected number of incorrect groups in the last t value positions. For $d = 1$, the effective strength is $(C_{R^{nw}})^l$, where incorrect group locations correspond to incorrect k_i bits are simply flipped. For $d = 2$, the effective strength is $(3C_{R^{nw}})^l$, where incorrect group locations must be permuted to examine the other possible κ_A, κ_B values. For $d \geq 3$ a first pass finds the κ_A, κ_B terms while subsequent passes probe up to $d-2$ nonlinear terms to extract keys $k_i, ..., k_{d+i-3}$. After the nonlinear terms are probed, the k_{d+i-2}, k_{d+i-1} keys are extracted from the κ terms obtained in the first pass. The effective strength is then approximately $3(C_{R^{nw}}^{d-1})^l$, although once the m keys are recovered, additional nonlinear terms that might be present are not examined.

The approach differs for SPECK which performs at most two passes per loop. For $d = 1$, a single pass is performed with keys extracted from κ_A giving the same effort as for SIMON $d = 1$. For $d = 2$, a single pass is performed where the data is grouped for κ_A, κ_B and the two key values contained in the single carry term. For $d \geq 3$, the first pass performs a number of attacks, A_C^*, where $A_C \leq A_C^* \leq n$ as to obtain the carry chain relationships. This estimation takes into account the additional number of attacks $(A_C^* - A_C)$ that must be performed given that some attacks may fail due to incorrect group recovery. Unlike SIMON where the incorrectly recovered group locations were permuted to examine the other groups, this is not feasible with SPECK due to the group size. However, since the attack density is sparse, and since the recovered chains give many different measurements of key relationships, it is expected that the incorrectly recovered groups can be identified through inconsistent relationships and then discarded, with only chains from attacks which provide consistent relationships being used. In the average case a minimum chain length of 3 (for the $d - 2$ chain) is used and for the worst case a minimum chain length of 1 is used. The best, average, and worst cases all include the brute force effort required to make use of the obtained key relationship information from the carry chains.

	$R^{\frac{3}{4}n}$							R^{n-1}							R^n	
d	Enc x	Bytes	B	$A1\sigma$	$A2\sigma$	$W1\sigma$	$W2\sigma$	Enc x	Bytes	B	$A1\sigma$	$A2\sigma$	$W1\sigma$	$W2\sigma$	Enc x	Bytes
SIMON 64/128 MSP $n = 32, m = 4$																
11	804	6KB	0.0	63.5	64.0	111.3	121.0	4351	34KB	0.0	20.8	26.2	36.2	49.7	5500	44KB
SIMON 128/256 MSP $n = 64, m = 4$																
1	804	12KB	0.0	126.6	127.9	214.7	228.4	4351	69KB	0.0	28.4	37.8	44.1	61.7	5500	88KB
SIMON 64/128 PIC $n = 32, m = 4$																
1	75	600B	0.0	63.5	64.0	111.3	121.0	673	5KB	0.0	20.8	26.2	36.2	49.7	930	7KB
SIMON 128/256 PIC $n = 64, m = 4$																
1	75	1KB	0.0	126.6	127.9	214.7	228.4	673	10KB	0.0	28.4	37.8	44.1	61.7	930	14KB
PUR SIMON 64/128 Spartan $n = 32, m = 4$																
1	284	2KB	0.0	63.5	64.0	111.3	121.0	3325	26KB	0.0	20.8	26.2	36.2	49.7	1E4	112KB
2	504	4KB	25.4	34.9	35.2	58.8	63.7	5911	47KB	3.2	13.6	16.2	21.3	28.0	3E4	200KB
3	2019	16KB	25.4	66.7	67.2	114.4	124.2	2E4	189KB	3.2	24.0	29.3	39.4	52.9	1E5	802KB
4	1E4	93KB	12.7	49.2	49.6	85.0	92.3	1E5	1MB	1.6	17.2	21.2	28.7	38.9	6E5	4MB
5	1E5	1MB	12.7	65.1	65.6	112.9	122.6	2E6	12MB	1.6	22.4	27.7	37.8	51.3	6E6	51MB
6	9E5	7MB	12.7	81.0	81.6	140.7	152.9	1E7	86MB	1.6	27.6	34.3	46.8	63.7	5E7	365MB
7	5E7	381MB	12.7	81.0	81.6	140.7	152.9	6E8	4GB	1.6	27.6	34.3	46.8	63.7	2E9	18GB
8	9E8	7GB	12.7	81.0	81.6	140.7	152.9	1E10	88GB	1.6	27.6	34.3	46.8	63.7	5E10	373GB
9	3E10	271GB	12.7	81.0	81.6	140.7	152.9	4E11	3TB	1.6	27.6	34.3	46.8	63.7	2E12	13TB
10	3E11	2TB	12.7	81.0	81.6	140.7	152.9	3E12	25TB	1.6	27.6	34.3	46.8	63.7	1E13	107TB
11	6E13	493TB	12.7	81.0	81.6	140.7	152.9	7E14	5PB	1.6	27.6	34.3	46.8	63.7	3E15	24PB
PUR SIMON 128/256 Spartan $n = 64, m = 4$																
1	314	5KB	0.0	126.6	127.9	214.7	228.4	6704	107KB	0.0	28.4	37.8	44.1	61.7	1E4	200KB
2	558	8KB	50.7	66.5	67.1	110.5	117.4	1E4	190KB	3.2	17.4	22.1	25.2	34.0	2E4	355KB
3	2232	35KB	50.7	129.8	131.1	217.9	231.6	5E4	762KB	3.2	31.6	40.9	47.3	64.8	9E4	1MB
4	1E4	205KB	25.4	96.5	97.5	162.6	172.9	3E5	4MB	1.6	22.9	29.9	34.7	47.8	5E5	8MB
5	1E5	2MB	25.4	128.2	129.5	216.3	230.0	3E6	48MB	1.6	30.0	39.4	45.7	63.3	6E6	91MB
6	1E6	16MB	25.4	159.8	161.4	270.0	287.1	2E7	347MB	1.6	37.1	48.8	56.7	78.7	4E7	647MB
7	5E7	842MB	25.4	159.8	161.4	270.0	287.1	1E9	17GB	1.6	37.1	48.8	56.7	78.7	2E9	33GB
8	1E9	16GB	25.4	159.8	161.4	270.0	287.1	2E10	355GB	1.6	37.1	48.8	56.7	78.7	4E10	662GB
9	4E10	599GB	25.4	159.8	161.4	270.0	287.1	8E11	12TB	1.6	37.1	48.8	56.7	78.7	1E12	23TB
10	3E11	4TB	25.4	159.8	161.4	270.0	287.1	6E12	102TB	1.6	37.1	48.8	56.7	78.7	1E13	190TB
11	7E13	1PB	25.4	159.8	161.4	270.0	287.1	1E15	23PB	1.6	37.1	48.8	56.7	78.7	3E15	43PB

Table 11: The suggested minimum rekeying frequency is presented for SIMON implementations along with the corresponding amount of data that can be encrypted with a single key before enough random plaintexts have been accumulated to correctly recovery at least R^{n_c} of the groups in each DGPA $\delta = 2$ attack for each power measurement examined with $n_c = \frac{3}{4}n, n-1, n$. The inability to fully recover the key is simply due to the limited statistics of the sample size. The expected number of encryptions, x, required to reach the desired recovery rate is given along with the corresponding data sample size in bytes. The best case, B, for all R^{n_c} finds the incorrectly recovered group locations on the first attempt. The average case, A, assumes that incorrect t values are in the lower half of all t values for $R^{\frac{3}{4}n}$ and in the lower quarter for R^{n-1}. The worst case, W, assumes that the location of the incorrectly recovered groups is found on the last attempt with all incorrect group t values being larger than all correct group t values. Average and Worst cases are presented for 1σ and 2σ confidence levels, where σ reflects the standard deviation from the expected number of incorrect groups for the number of encryptions examined, taken as $\sigma = 3$ bits for $R^{\frac{3}{4}n}$ and $\sigma = 1$ bit for R^{n-1}. Best, Average and Worst case results are presented as the effective number of key bits, n_e, which should be compared to mn. Note that in some worst case scenarios, $n_e > mn$, meaning the result is worst than brute forcing the mn key bits due to a total of more than m DGPA passes being required. Data size suffixes are in increments of 10^3 bytes and are B, KB, MB, GB, TB, and PB.

	$R^{\frac{3}{4}n}$							R^{n-1}							R^n	
d	Enc x	Bytes	B	$A1\sigma$	$A2\sigma$	$W1\sigma$	$W2\sigma$	Enc x	Bytes	B	$A1\sigma$	$A2\sigma$	$W1\sigma$	$W2\sigma$	Enc x	Bytes
SPECK 64/128 MSP $n = 32, m = 4$																
1	254	2KB	0.0	68.5	69.0	116.3	126.0	2538	20KB	0.0	25.8	31.2	41.2	54.7	5000	40KB
SPECK 128/256 MSP $n = 64, m = 4$																
1	254	4KB	0.0	132.6	133.9	220.7	234.4	2538	40KB	0.0	34.4	43.8	50.1	67.7	5000	80KB
SPECK 64/128 PIC $n = 32, m = 4$																
1	57	456B	0.0	68.5	69.0	116.3	126.0	593	4KB	0.0	25.8	31.2	41.2	54.7	1300	10KB
SPECK 128/256 PIC $n = 64, m = 4$																
1	57	912B	0.0	132.6	133.9	220.7	234.4	593	9KB	0.0	34.4	43.8	50.1	67.7	1300	20KB
PUR SPECK 64/128 Spartan $n = 32, m = 4$																
1	10	80B	0.0	68.5	69.0	116.3	126.0	1387	11KB	0.0	25.8	31.2	41.2	54.7	2E4	132KB
2	20	160B	25.4	74.9	75.3	122.6	132.4	2774	22KB	3.2	32.2	37.5	47.5	61.0	3E4	264KB
3	640	5KB	28.7	93.3	95.4	131.3	140.4	9E4	710KB	17.6	58.4	63.3	72.5	83.5	1E6	8MB
4	5E6	41MB	28.7	55.4	56.5	80.3	86.1	7E8	5GB	17.6	32.6	35.7	41.6	48.8	9E9	69GB
5	2E13	175TB	36.7	47.5	48.5	60.5	63.9	3E15	24PB	25.6	35.4	37.2	40.5	44.3	4E16	291PB
6	4E20	2ZB	32.0	38.0	38.9	39.1	40.0	5E22	409ZB	32.0	36.6	37.0	37.9	38.3	6E23	4YB
PUR SPECK 128/256 Spartan $n = 64, m = 4$																
1	10	160B	0.0	132.6	133.9	220.7	234.4	3653	58KB	0.0	34.4	43.8	50.1	67.7	3E4	523KB
2	20	320B	50.7	138.9	140.2	227.1	240.8	7306	116KB	3.2	40.7	50.1	56.4	74.0	7E4	1MB
3	640	10KB	41.4	142.6	145.4	210.6	222.7	2E5	3MB	17.6	66.1	74.0	80.2	94.2	2E6	33MB
4	5E6	83MB	41.4	87.9	89.5	132.9	140.7	2E9	30GB	17.6	37.4	42.5	46.4	55.6	2E10	274GB
5	7E14	11PB	49.4	64.3	65.5	87.2	91.6	3E17	4EB	25.6	38.3	41.0	43.4	48.2	2E18	36EB
6	2E26	3E27B	32.0	39.0	39.9	40.0	40.9	7E28	1E30B	32.0	37.6	38.0	38.8	39.2	6E29	1E31B

Table 12: The suggested minimum rekeying frequency is presented for SPECK implementations along with the corresponding amount of data that can be encrypted with a single key before enough random plaintexts have been accumulated to correctly recovery at least R^{n_c} of the groups in each DGPA $\delta = 2$ attack for each power measurement examined with $n_c = \frac{3}{4}n, n-1, n$. The inability to fully recover the key is simply due to the limited statistics of the sample size. The expected number of encryptions, x, required to reach the desired recovery rate is given along with the corresponding data sample size in bytes. The best case, B, for all R^{n_c} finds the correct key on the first attempt. The average case, A, assumes that incorrect t values are in the lower half of all t values for $R^{\frac{3}{4}n}$ and in the lower quarter for R^{n-1}. The worst case, W, assumes that the location of the incorrectly recovered groups is found on the last attempt with all incorrect group t values being larger than all correct group t values. Average and Worst cases are presented for 1σ and 2σ confidences, where σ reflects the standard deviation from the expected number of incorrect groups for the number of encryptions examined, taken as $\sigma = 3$ bits for $R^{\frac{3}{4}n}$ and $\sigma = 1$ bit for R^{n-1}. Best, Average and Worst case results are presented as the effective number of key bits which should be compared to mn. Data size suffixes are in increments of 10^3 bytes and are B, KB, MB, GB, TB, PB, EB, ZB, and YB.

10.1 Continuous Approximation

An additional approach was taken to examine the key recovery effort as a continuous function dependent on acquired data size for various d. The potential values of the DGPA $\delta = 2$ equation pair as either having zero, one (from either of the equations), or two bits flipped in the result were modeled as Gaussian functions for each of the four possible outcomes. Corresponding $t_{2,0}$-test value distributions were derived as those obtained from the possible selections of correct or incorrect groupings.

The Gaussian distributions included a fixed systematic error contribution and a statistical error contribution proportional to $\frac{1}{\sqrt{x}}$, with x being the number of encryptions performed with random plaintext. The parameters describing these error contributions were selected so that the cases of 2-bit AON, Coupled Normal Hierarchy, Coupled Inverted Hierarchy, and G_1 all closely matched the fit parameters of table 3. It was then possible to determine the probability distributions for G_2, G_1, G_0 comprising the selected groups having the largest t-test values and the expected location of the maximum t-test value corresponding to an incorrectly recovered group relative to the correctly recovered groups as a function of x.

The effort for each step of the attack was calculated as the combined probability of finding the locations of each of the different types of incorrectly recovered groups along with the cost of permuting the appropriate bits to examine the other possible groups. For instance, if a G_1 group is encountered, then one of the two equation results is incorrect resulting in two candidates for the G_2 solution which must be examined. However, if a G_0 group is encountered, then both equation results are wrong and the alternative to obtain G_2 is examined in a single step.

The results of this approach are presented both for SIMON 64/128 and SPECK 64/128 in figure 6. As most modes of block cipher operation specify a limit of 2^n encryptions to be performed with a given key to ensure security for a block size of $2n$, it is only necessary to preclude implementations which can feasibly make use of a sample size less than that limit. A practical limit on the effort required to perform the attack is chosen to be 80-bits of work. It is seen that DGPA attacks on SIMON 64/128 $d \geq 9$ and SPECK 64/128 $d \geq 5$ are not feasible.

This more detailed continuous approximation approach is found to provide results consistent with the previously described estimation.

11 Computational Cost

There are multiple methods of implementing the attacks described in this paper. Divided into two approaches, an implementation can either gather a quantity of data which is then processed, or the data can be processed and discarded as it is accumulated. This discussion assumes an attack on random plaintext, but an attack on random ciphertext follows the same approach. If the attacker is able to encrypt specifically generated data, then it may not be necessary to store the plaintext if generated with a known pseudo random function and seed.

Calculations described in this section assume 16-bit (2 byte) data values for power measurements and associated values (sums, averages, standard deviations), 4 byte integers for group statistical bin event counters, and $\frac{2n}{8}$ bytes for plaintext or ciphertext storage.

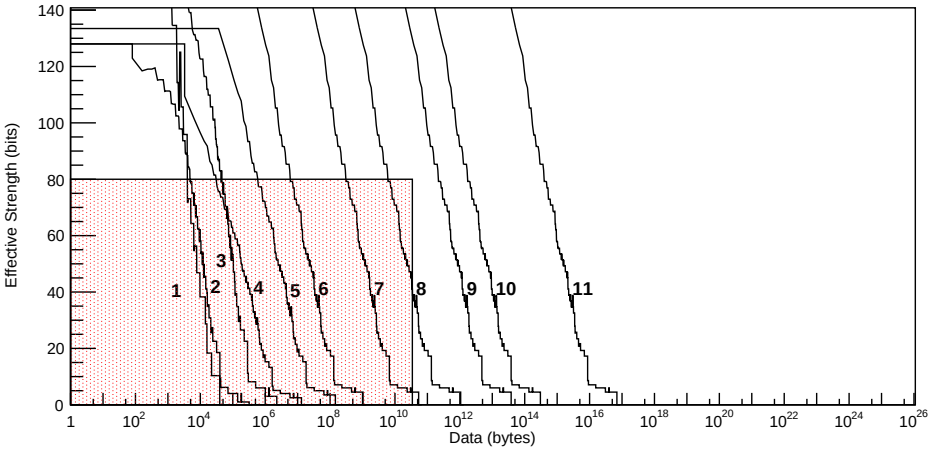
In the Gather then Process (GtP) method, a total of x random plaintext are recorded along with the corresponding $\lceil \frac{m}{d} \rceil$ data points corresponding to the first $\lceil \frac{m}{d} \rceil$ encryption cycles. The data storage requirement is then $\frac{2n}{8} + 2\lceil \frac{m}{d} \rceil$ bytes per encryption. Once a preset number of encryptions have been recorded, the processing is performed over the x entries where for each bit of random plaintext and key hypothesis an average and standard deviation of the power for all DGPA groups are calculated. For $d > 1$, multiple passes may be required as keys are partially recovered in each pass. Each pass requires temporary space (reusable with each pass) to store the average/standard deviation values for each bit and key

hypothesis along with the number of recorded entries for each bin, resulting $6n$ bytes per key hypothesis. In the case of two bit DGPA ($\delta = 2$), at most 8 groupings are required for the key hypothesis resulting in a requirement of $8 \cdot 6n$ bytes of temporary space. The nonlinear attacks examine a single group with two groupings using the found κ values. Actual attacks suitable for use in scaling the results of tables 13 and 14 are obtained using a non-optimized GtP method implemented in JAVA using `BigInteger` operations on single core CPU at 2.8GHz which recovered key values from 50k plaintext/power data events in 12, 21, 14, and 20 seconds for SIMON 64/128 $d = 1$, SIMON 128/256 $d = 1$, SPECK 64/128 $d = 1$, and SPECK 128/256 $d = 1$, including I/O time. An optimized JAVA version of SIMON 64/128 key recovery measuring only computation time of the recovery performs with an average time per c per n per x of 1.7E−7s for storing and 3.1E−10s for recovery, which are scale factors suitable for multiplying the complexities in tables 13 and 14.

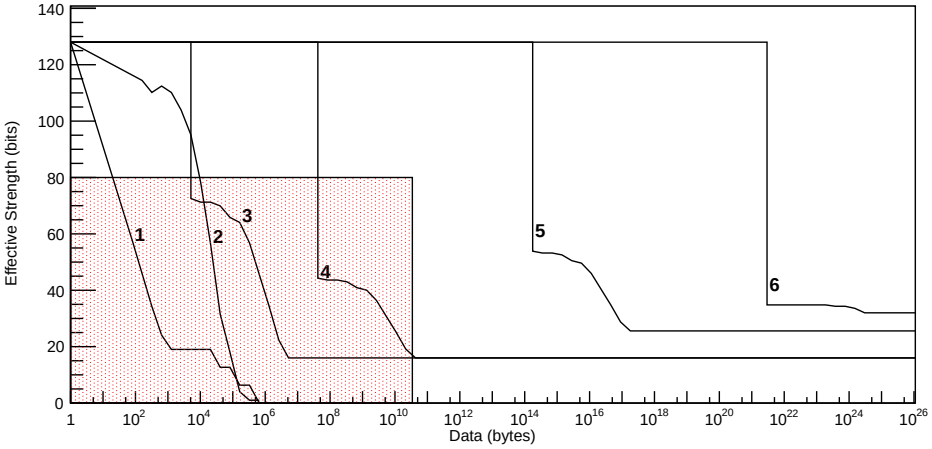
Another approach is the Process and Discard (PaD) method where data for each encryption is processed as it is obtained and then discarded. The storage savings obtained with this method is offset by the requirement that all possible key hypothesis for each DGPA attack must be accounted for at the onset. The overall computational complexity is similar where the group statistic calculation is simply relocated from after data acquisition (for GtP) to during data acquisition (for PaD). However, this results in a throttling effect where the rate of data acquisition can be limited due to the cost of storing encryption event data. Data obtained at a faster rate can either be queued or discarded as required.

Computational and resource cost calculations for both methods are performed for $n = 32, 64$, $m = 4$ implementations and are presented in table 13 for SIMON and table 14 for SPECK. The limits where the attack methods begin to become unfeasible are examined, which assuming that a single core machine might be limited to 2GB RAM, 1 TB disk, 1GFlop/s and assuming that a large super computer might be limited to 1 PB RAM, 1 EB storage, 50 PFlop/s. A limit is deemed unfeasible if it cannot fit on the constraints of the platform or if it is expected to take more than a year of processing. GtP data can be written to disk while GtP temporary space and PaD data should be held in RAM. The SIMON PaD method can be performed on either platform for any d and is only throttled in the rate of data collection. If all of the data is available at once, then SIMON 64/128 becomes unfeasible for the single core at $d = 10$ and for the super computer at $d = 14$. The SIMON GtP method is unfeasible for both platforms at the same or smaller d. The PaD method for SPECK 64/128 becomes unfeasible on the single core at $d = 5$ and on the super computer at $d = 6$ from computational and memory constraints. The SPECK GtP method is unfeasible for both platforms at the same or smaller d.

In the simplest cases, the requirements are small enough that an embedded microcontroller operating at about 1MHz along with a peak sensing ADC circuit may be situated in the field which carries out an attack and which transmits the found key value only after verifying that it is correct with a known plaintext/ciphertext pair. The SIMON 64/128 d=1/2 GtP requirement is 226/302 KB with less than 4/7 seconds to perform a recovery. The SIMON 64/128 d=1/2 PaD requirement is 3.6/1.1 KB with total data storage times of 7/4 seconds and less than 1 second to perform a recovery. The SPECK 64/128 d=1/2 GtP requirement is 265/397 KB with less than 40 seconds to perform a recovery.



(a) SIMON 64/128



(b) SPECK 64/128

Fig. 6: Results from an estimation of the work required to recover the key of SIMON 64/128 (top) and SPECK 64/128 (bottom) using a Gaussian model of the distribution of bit flips and t-test values corresponding to the correctly chosen G_2 and incorrectly chosen G_1, G_0 as a function of the number of encryptions of random plaintext captured. The model is calibrated for the fit parameters obtained from $d = 1, 2$ Spartan recoveries of SIMON 64/128 and SPECK 64/128 and represents the average case at 1σ for the fraction of expected correct group recoveries as a function of number of encryptions. The number of plaintexts is presented as the total number of data bytes encrypted (x-axis) and the effective strength of the implementation is presented as the number of bits of effort required (y-axis) and should be compared to the expected key strength of $nm = 128$ bits. Lines are drawn and labeled for different d. An exclusion region is shown (shaded area) representing less than 80-bits of work and less than 2^n encryptions. Implementations which pass through this region can be feasibly attacked. It is seen that attacks against implementations of SIMON 64/128 with $d \geq 9$ and implementations of SPECK 64/128 with $d \geq 5$ are not feasible. Note that for SIMON, $d = 1, 3$ or $d \geq 5$ require at least m DGPA passes and may exceed the brute force effort of 128 bits with low statistics whereas for SPECK, DGPA with $d \geq 3$ cannot be performed until a sufficient number of samples have been obtained to populate each of the various groups.

							Storage Bytes				Complexity		
n	m	d	x	c	p	g	Data	Temp	Key	Total	Store	Store x	Recovery
Gather then Process (GtP) SIMON 64/128													
32	4	1	1.41E4	4	1	2	2.26E5	512	16	2.26E5	16	2.26E5	3.61E6
32	4	2	2.51E4	2	1	4	3.01E5	1024	16	3.02E5	12	3.01E5	6.42E6
32	4	3	1.00E5	2	2	8	1.20E6	2048	32	1.21E6	12	1.20E6	7.70E7
32	4	4	5.78E5	1	3	16	5.78E6	4096	28	5.78E6	10	5.78E6	7.39E8
32	4	5	6.42E6	1	4	16	6.42E7	4096	28	6.42E7	10	6.42E7	1.15E10
32	4	6	4.56E7	1	5	16	4.56E8	4096	28	4.56E8	10	4.56E8	1.05E11
32	4	7	2.37E9	1	5	16	2.37E10	4096	28	2.37E10	10	2.37E10	5.45E12
32	4	8	4.67E10	1	5	16	4.67E11	4096	28	4.67E11	10	4.67E11	1.08E14
32	4	9	1.68E12	1	5	16	1.68E13	4096	28	1.68E13	10	1.68E13	3.88E15
32	4	10	1.34E13	1	5	16	1.34E14	4096	28	1.34E14	10	1.34E14	3.09E16
32	4	11	3.06E15	1	5	16	3.06E16	4096	28	3.06E16	10	3.06E16	7.06E18
Gather then Process (GtP) SIMON 128/256													
64	4	1	1.25E4	4	1	2	3.00E5	1024	32	3.01E5	24	3.00E5	6.40E6
64	4	2	2.22E4	2	1	4	4.44E5	2048	32	4.47E5	20	4.44E5	1.14E7
64	4	3	8.89E4	2	2	8	1.78E6	4096	64	1.78E6	20	1.78E6	1.37E8
Process and Discard (PaD) SIMON 64/128													
32	4	1	1.41E4	4	1	14	3584	0	16	3600	448	6.32E6	462
32	4	2	2.51E4	2	1	4	1024	0	16	1040	128	3.21E6	132
32	4	3	1.00E5	2	2	12	3072	0	32	3104	384	3.85E7	396
32	4	4	5.78E5	1	3	24	6144	0	28	6172	768	4.44E8	792
32	4	5	6.42E6	1	4	152	3.89E4	0	28	3.89E4	4864	3.12E10	5016
32	4	6	4.56E7	1	5	152	3.89E4	0	28	3.89E4	4864	2.22E11	5016
32	4	7	2.37E9	1	5	152	3.89E4	0	28	3.89E4	4864	1.15E13	5016
32	4	8	4.67E10	1	5	152	3.89E4	0	28	3.89E4	4864	2.27E14	5016
32	4	9	1.68E12	1	5	152	3.89E4	0	28	3.89E4	4864	8.18E15	5016
32	4	10	1.34E13	1	5	152	3.89E4	0	28	3.89E4	4864	6.52E16	5016
32	4	11	3.06E15	1	5	152	3.89E4	0	28	3.89E4	4864	1.49E19	5016
Process and Discard (PaD) SIMON 128/256													
64	4	1	1.25E4	4	1	14	7168	0	32	7200	896	1.12E7	910
64	4	2	2.22E4	2	1	4	2048	0	32	2080	256	5.69E6	260
64	4	3	8.89E4	2	2	12	6144	0	64	6208	768	6.83E7	780

Table 13: The estimated approximate required storage and computational resources to perform a full key recovery DGPA attack on PUR SIMON implementations for various block sizes, $2n$, key sizes, mn, and rounds computed per clock cycle, d, assuming x events are required for full key recovery as obtained from scaled Spartan data. The DGPA requires at most p passes for each of c power measurements used in the attack. Comparisons are made between the Gather then Process (GtP) and Process and Discard (PaD) methods, where GtP stores all plaintext and power data and then performs a single key extraction operation and where the PaD accumulates and stores group statistic information as to calculate average, standard deviation, and event count groups necessary for the computation from plaintext and power data which is then discarded. The storage requirement in bytes includes that for the data (plaintext/power values for GtP and group statistic values for PaD), temporary space (to calculate group statistic values for GtP), and key storage space to hold accumulated probably keys and associated κ values. Complexity is given as the number of operations and should be used as a scaling factor to estimate time cost from experimental results. Complexity to store a single plaintext/power measurement and to store the total x necessary samples for complete key recovery is given. The recovery complexity is the cost of performing statistic calculations for each group value permutation and finding keys (for GtP), and for finding keys from already calculated group statistics (for PaD). Storage Bytes and Recovery Complexity (and Storage Complexity for PaD) scale by n, Storage Complexity (for GtP) includes the addition of a $\frac{n}{8}$ term, and Storage Bytes and Recovery for GtP scales by x, allowing for the presented results to be extended to other d and n.

d	x	c	p	n_{g_κ}	n_{g_C}	$g_\kappa g_C$	A_c	b	Storage Bytes Data	Temp	Key	Total	Complexity Store	Store x	Rec A	Rec B
Gather then Process (GtP) SPECK 64/128 $n = 32, m = 4$																
1	1.66E4	4	1	2	0	4	0	0	2.65E5	32	16	2.65E5	16	2.65E5	0	3.39E7
2	3.31E4	2	1	2	2	16	0	0	3.97E5	128	24	3.97E5	12	3.97E5	0	3.39E7
3	1.06E6	2	2	3	6	512	16	16	1.27E7	4096	40	1.27E7	12	1.27E7	1.22E9	5.56E11
4	8.68E9	1	2	3	19	4.19E6	8	16	8.68E10	3.36E7	28	8.68E10	10	8.68E10	1.06E13	5.83E17
5	3.64E16	1	2	3	41	1.76E13	8	24	3.64E17	1.41E14	28	3.64E17	10	3.64E17	9.56E19	6.26E26
6	6.11E23	1	2	3	65	2.95E20	8	32	6.11E24	2.36E21	28	6.11E24	10	6.11E24	2.54E27	2.79E11
Gather then Process (GtP) SPECK 128/256 $n = 64, m = 4$																
1	3.27E4	4	1	2	0	4	0	0	7.86E5	32	32	7.86E5	24	7.86E5	0	1.34E8
2	6.55E4	2	1	2	2	16	0	0	1.31E6	128	48	1.31E6	20	1.31E6	0	1.34E8
3	2.10E6	2	2	3	6	512	24	16	4.19E7	4096	80	4.19E7	20	4.19E7	3.62E9	2.20E12
4	1.72E10	1	2	3	19	4.19E6	12	16	3.09E11	3.36E7	56	3.09E11	18	3.09E11	3.13E13	2.30E18
5	2.30E18	1	2	3	46	5.63E14	12	24	4.15E19	4.50E15	56	4.15E19	18	4.15E19	1.02E22	7.92E28
6	6.33E29	1	2	3	84	1.55E26	12	32	1.14E31	1.24E27	56	1.14E31	18	1.14E31	5.11E33	3.61E11
Process and Discard (PaD) SPECK 64/128 $n = 32, m = 4$																
1	1.66E4	4	1	2	0	4	0	0	1.10E12	0	16	1.10E12	1.37E11	2.28E15	0	512
2	3.31E4	2	1	2	2	16	0	0	4.20E6	0	24	4.20E6	2.62E5	8.69E9	0	768
3	1.06E6	2	2	3	6	512	16	16	3.52E13	0	40	3.52E13	4.12E11	4.37E17	8.80E12	1.68E7
4	8.68E9	1	2	3	19	4.19E6	8	16	1.07E9	0	28	1.07E9	4864	4.22E13	3.36E7	1.68E7
5	3.64E16	1	2	3	41	1.76E13	8	24	4.50E15	0	28	4.50E15	1.05E4	3.82E20	1.41E14	4.29E9
6	6.11E23	1	2	3	65	2.95E20	8	32	1.89E22	0	28	1.89E22	4160	2.54E27	2.36E21	1.10E12
Process and Discard (PaD) SPECK 128/256 $n = 64, m = 4$																
1	3.27E4	4	1	2	0	4	0	0	2.20E12	0	32	2.20E12	2.75E11	9.00E15	0	1024
2	6.55E4	2	1	2	2	16	0	0	8.40E6	0	48	8.40E6	5.25E5	3.44E10	0	1536
3	2.10E6	2	2	3	6	512	24	16	5.28E13	0	80	5.28E13	6.18E11	1.30E18	1.32E13	3.36E7
4	1.72E10	1	2	3	19	4.19E6	12	16	2.15E9	0	56	2.15E9	9728	1.67E14	5.03E7	3.36E7
5	2.30E18	1	2	3	46	5.63E14	12	24	2.88E17	0	56	2.88E17	2.36E4	5.43E22	6.76E15	8.59E9
6	6.33E29	1	2	3	84	1.55E26	12	32	1.49E28	0	56	1.49E28	8064	5.11E33	1.86E27	2.20E12

Table 14: The estimated approximate required storage and computational resources to perform a full key recovery DGPA attack on PUR SPECK implementations for various block sizes, $2n$, key sizes, mn, and rounds computed per clock cycle, d, assuming x events are required for full key recovery as obtained from scaled Spartan data. The DGPA requires at most p passes for each of c power measurements used in the attack. For $d \geq 2$, carry chains are obtained from A_c attacks which form n_{g_C} groups and which require b bits to be permuted to obtain the values from the carry chains. The number of groups associated with the κ terms are n_{g_κ}. The total number of grouping permutations is $g_\kappa g_C$. The process is repeated for c power measurements. Comparisons are made between the Gather then Process (GtP) and Process and Discard (PaD) methods. The storage requirement in bytes includes that for the data (plaintext/power values for GtP and group statistic values for PaD), temporary space (to calculate group statistic values for GtP), and key storage space to hold accumulated probably keys and associated κ values. Complexity is given as the number of operations and should be used as a scaling factor to estimate time cost from experimental results. Complexity to store a single plaintext/power measurement and to store the total x necessary samples for complete key recovery is given. The recovery complexity is the cost of performing statistic calculations for each group value permutation and finding keys (for GtP), and for finding keys from already calculated group statistics (for PaD). The first key recovery step determines the key relationship (Rec A) from the carry chains for $d > 2$. The second key recovery step (Rec B) determines key values by permuting the b bits which when used with the chains recover $d-2$ keys for $d > 2$. If keys are to be recovered from the linear key terms in a $d = 2$ type pass, that process is also performed in this step. For $c > 1$, the PaD method includes the accounting for all possible key values from prior cycles in both storage and computational complexity which is G_t from table 9.

12 Implementing $d > 1$

The value of d chosen for an implementation will be made to provide the necessary minimum level of security for an application while satisfying hardware and power constraints. When implementing SIMON or SPECK with $d > 1$, there are some considerations to take into account when d does not divide T. A d' with $d' > d$ may be chosen such that d' divides T only if sufficient resources are available. In keeping to the standard, simply performing additional rounds so that an integer multiple of d performs a total of T' rounds with $T' > T$ would not be a suitable solution.

An implementation with d' rounds where $d' > d$ can use $2n$ 2:1 multiplexers (MUXs) to select results from a depth of d_0 for the first c_0 cycles and then from a depth of d_1 for the last c_1 cycles where $d_0c_0 + d_1c_1 = T$. This requires both $d_0 \geq d$ and $d_1 \geq d$ as to provide the necessary protection. The minimum of d_0, d_1 is denoted d_m and the maximum of d_0, d_1 is d'. If $c_1 \geq 2$ then the key schedule algorithm which is carried out in parallel with the rounds requires the use of nm additional MUXs to select input values for subsequent cycles. All possible combinations of d_0, d_1 were examined for SIMON, requiring $d \geq 9$, and for SPECK, requiring $d \geq 5$, and the combinations with the minimum resulting d' are presented in table 15. As a MUX can generate a signal on a transition which differs when the inputs are either the same or different values, it is vital to ensure that the MUX inputs correspond to the round logic state of an inner cycle during the MUX transition. If the MUX transition occurs while the inputs to the MUX correspond to the computation for the final cycle which is used to produce the ciphertext, then a signal related to the ciphertext at a distance of $d' - d_m$ rounds is created. The proper MUX transition can be ensured either through a careful layout and timing analysis or through the use of two-stage PUR registers which can withhold the updating of the round logic with the previous cycle result until an additional clock is issued during which time the MUXs transition while the round logic state remains constant. Any signal generated in this proper way will correspond to a depth of $c_1d_1 + d' - d_m$ and will be less useful to the attacker than a d_m transition.

The result at T rounds may also be directly extracted from the round logic in the last clock cycle into a different destination register without the use of MUXs when computing c cycles of d rounds where $cd > T$, although this approach requires careful treatment. The number of extra rounds computed at d is $\eta = d - (T \mod d)$, resulting in the $\mathbf{X}, \mathbf{Y}$ PUR registers holding the values of $\mathbf{X}_{T+\eta-d}, \mathbf{Y}_{T+\eta-d}$ in the penultimate and $\mathbf{X}_{T+\eta}, \mathbf{Y}_{T+\eta}$ in the final clock cycles. The PUR registers should only be updated in the final cycle if $2\eta > d$ as to maximize the effort of a reduced round cryptanalysis in the event where the final PUR register values can be determined. In any case, the contents of the PUR registers must be sanitized prior to overwriting with the plaintext of the next encryption, otherwise signals at a distance of less than d rounds from the ciphertext may be available. The sanitization would require overwriting the final value of the PUR registers with a random value and then clearing those registers prior to loading the next plaintext. Due to the cost and difficulty of implementing the random value generator, this approach should be avoided.

SIMON													
$2n$	nm	T	d_0	c_0	d_1	c_1	d'	c_{tot}	$A_\propto$	$E_\propto$	$E_\propto/2n$	$W_\propto$	$W_\propto/2n$
32	64	32	11	2	10	1	11	3	176	528	16	16	0.5
48	72	36	9	4	-	-	9	4	216	864	18	0	0
48	96	36	9	4	-	-	9	4	216	864	18	0	0
64	96	42	11	3	9	1	11	4	352	1408	22	64	1
64	128	44	11	4	-	-	11	4	352	1408	22	0	0
96	96	52	10	4	12	1	12	5	576	2880	30	384	4
96	144	54	9	6	-	-	9	6	432	2592	27	0	0
128	128	68	11	5	13	1	13	6	832	4992	39	640	5
128	128	68	10	5	9	2	10	7	640	4480	35	128	1
128	192	69	10	6	9	1	10	7	640	4480	35	64	0.5
128	256	72	9	8	-	-	9	8	576	4608	36	0	0

SPECK													
$2n$	nm	T	d_0	c_0	d_1	c_1	d'	c_{tot}	$A_\propto$	$E_\propto$	$E_\propto/2n$	$W_\propto$	$W_\propto/2n$
32	64	22	5	3	7	1	7	4	112	448	14	96	3
32	64	22	6	2	5	2	6	4	96	384	12	32	1
48	72	22	5	3	7	1	7	4	168	672	14	144	3
48	72	22	6	2	5	2	6	4	144	576	12	48	1
48	96	23	6	3	5	1	6	4	144	576	12	24	0.5
64	96	26	5	4	6	1	6	5	192	960	15	128	2
64	128	27	5	4	7	1	7	5	224	1120	17	256	4
96	96	28	7	4	-	-	7	4	336	1344	14	0	0
96	96	28	6	3	5	2	6	5	288	1440	15	96	1
96	144	29	6	4	5	1	6	5	288	1440	15	48	0.5
128	128	32	5	5	7	1	7	6	448	2688	21	640	5
128	192	33	7	4	5	1	7	5	448	2240	17	128	1
128	256	34	7	4	6	1	7	5	448	2240	17	64	0.5
128	256	34	6	4	5	2	6	6	384	2304	18	128	1

Table 15: Area optimized choices for 2:1 multiplexer (MUX) locations are presented for computing exactly T rounds in PUR implementations of SIMON and SPECK with block size $2n$ and key size nm using round logic having a depth of d' which is greater than the minimum d required for security. Minimum d values of $d \geq 9$ and $d \geq 5$ are used for SIMON and SPECK, respectively. A total of $2n$ MUXs are used to select the output of the round logic at a depth of d_0 rounds for the first c_0 cycles and then at a depth of d_1 rounds for the final c_1 cycles for a total of $c_{tot} = c_0 + c_1$ cycles. Solutions with $c_1 \geq 2$ require an additional mn corresponding MUXs in the key schedule algorithm which is performed in parallel with the rounds of the encryption. Best cases for the classes of $c_1 \leq 1$ and $c_1 \geq 2$ are both provided. Scaling values comparing the area cost, $A_\propto = nd'$, approximate energy cost, $E_\propto = A_\propto c_{tot}$, approximate energy cost per bit encrypted, $E_\propto/2n$, approximate wasted energy, $W_\propto = E_\propto - nT$, and approximate wasted energy per bit encrypted, $W_\propto/2n$, are given. Note that the estimations on energy cost do not take into account the SPECK signal propagation of the addition carry bits. It is seen that in some cases the values of $A_\propto, E_\propto, E_\propto/2n, W_\propto, W_\propto/2n$ may be the same or reduced for larger m and/or n.

13 Other Attack Methods

Other methods of power analysis may be carried out on PUR SIMON or PUR SPECK implementations which differ in approach from the DGPA $\delta = 2$ method. Multiple bit PPAs have a general advantage over single bit PPAs from the aspect of an increased signal to noise ratio (SNR) when dealing with additive Gaussian noise on quantized power per bit register state change transitions. This increased SNR generally results in the use of fewer events to achieve the same level of key recovery. However, when additional bits are examined, the number of data partitions that must be made will generally stay the same or increase as more unknown key bits will be present countering the effect of the increased SNR.

In the unrealistic limit of zero noise, it is possible to examine the best case result of a 1-bit PPA or DPA compared to the DGPA $\delta = 2$ method. The number of groups that are formed under standard 1-bit partitioning based on unknown key bits and that are formed using a hybrid 1-bit partitioning using the SIMON linear approximation and SPECK carry chains described in this paper is compared to the DGPA $\delta = 2$ method in table 16. It is seen that the number of encryptions required for the standard partitioning based on unknown key bits alone far exceeds that of the DGPA $\delta = 2$. For SIMON, the 1-bit hybrid PPA method requires half of the partitions, thus reducing the number of encryptions required by a factor of $\frac{1}{\sqrt{2}}$, although it also requires additional passes for $d = 2, 3, 4$. This still results in an attack on SIMON 64/128 $d \geq 9$ requiring more encryptions then the limited $2n$-bit block cipher key lifespan of 2^n encryptions. For SPECK, the 1-bit hybrid PPA method reduces the number of required encryptions substantially although increases the number of attacks required to obtain the key carry chains. Using the bits of $\mathbf{X}$ for SPECK 64/128 $d = 5$, the number of partitions is 2^{29} and requiring a minimum of 10 encryptions per partition still puts the data requirement above the 2^{32} encryption limit. SPECK appears to require many fewer events when attacking $\mathbf{X}$ and almost the same when attacking $\mathbf{Y}$. However, when using only $\mathbf{X}$ in a 1-bit PPA, only one key bit, k_{d-1}, can be obtained from the linear term resulting in an increased brute force effort of $\min(m, d-1)$ terms. This makes $d = 3, 4, 5$ harder by an effort of 8 additional bits of work over the same d level DGPA $\delta = 2$. The C_{d-1} carry term only appears once and does not form a chain as does the C_{d-2} terms which form a chain of 2 terms, resulting in an increased attack count. At $d = 5$ the brute force effort is 2^{32} instead of 2^{24}, and the attack count is tripled. Still conservatively requiring a minimum of 10 entries per group puts the required data for the SPECK 64/128 hybrid 1-bit PPA attack using the X register at 43 GB and over 2^{33} encryptions, which are more encryptions than the key can safely encrypt.

It is also possible to perform multiple 1-bit PPA attacks separately to achieve different key hypothesis from which inconsistencies are used to identify bits that should be permuted. This results in essentially the gain which is achieved by coupling the equations and performing the DGPA $\delta = 2$ or another multi-bit PPA, although without the improved SNR.

Higher n-th order PPA attacks may be used in cases where information contained in the power consumption of n clock cycles is combined. Such methods are typically used to circumvent random bit masking countermeasures where the power from the mask application operation and the power from the use of the masked value can be combined to cancel the effect of the mask. In the PUR implementations of SIMON and SPECK described here, the combination of power samples from multiple clock cycles in a higher order attack could be used to obtain the equivalent power signals of $\mathbf{X}, \mathbf{Y}$ register overwrites at multiples of d. Such information is not expected to be more useful in a power analysis than a power analysis simply performed at d.

SIMON 64/128										
	1-bit PPA S				1-bit PPA H				DGPA $\delta = 2$	
d	X_g	Y_g	X_p	Y_p	X_g	Y_g	X_p	Y_p	$(XY)_g$	$(XY)_p$
1	2^1	2^1	4	4	2^1	2^1	4	4	2^1	4
2	2^3	2^{21}	2	1	2^1	2^1	4	5	2^2	2
3	2^7	2^{21}	2	1	2^1	2^1	5	5	2^2	4
4	2^{11}	2^{51}	1	1	2^1	2^1	4	5	2^2	3
5	2^{21}	2^{11}	1	1	2^1	2^1	5	4	2^2	4
6	2^{32}	2^{21}	1	1	2^1	2^1	5	5	2^2	5
7	2^{51}	2^{32}	1	1	2^1	2^1	5	5	2^2	5
8	2^{72}	2^{51}	1	1	2^1	2^1	5	5	2^2	5
9	2^{98}	2^{72}	1	1	2^1	2^1	5	5	2^2	5
10	2^{127}	2^{98}	1	1	2^1	2^1	5	5	2^2	5
11	2^{159}	2^{127}	1	1	2^1	2^1	5	5	2^2	5

SPECK 64/128														
	1-bit PPA S				1-bit PPA H						DGPA $\delta = 2$			
d	X_g	Y_g	X_p	Y_p	X_g	Y_g	X_p	Y_p	A_C	MES	$(XY)_g$	$(XY)_p$	A_C	MES
1	2^1	2^1	4	4	2^1	2^1	4	4	-	-	2^3	4	-	-
2	2^3	2^3	2	2	2^2	2^2	2	2	-	-	2^5	2	-	-
3	2^9	2^{11}	2	2	2^6	2^7	2	2	48	2^{24}	2^8	2	16	2^{16}
4	2^{23}	2^{30}	1	1	2^{15}	2^{19}	1	1	24	2^{24}	2^{21}	1	8	2^{16}
5	2^{44}	2^{59}	1	1	2^{30}	2^{40}	1	1	24	2^{32}	2^{43}	1	8	2^{24}
6	2^{69}	2^{91}	1	1	2^{48}	2^{64}	1	1	8	2^{32}	2^{67}	1	8	2^{32}

Table 16: Comparisons of other attack methods to DGPA $\delta = 2$. The number of partitions required when using 1-bit from either $\mathbf{X}$ or $\mathbf{Y}$ register overwrites is given at X_g, Y_g with partitions for the DGPA given as $(XY)_g$. Similarly, the number of passes required to extract all of the m key bits is given as $X_p, Y_p, (XY)_p$. Traditional standard (S) 1-bit PPA (including DPA) partitions the data accounting for the various permutations of the unknown key bits. The number of data partitions grows substantially with d for both SIMON and SPECK making the required amount of data unmanageable for larger d. The SPECK standard 1-bit PPA approach does not take into account the additional effort of carry bit accounting. A hybrid (H) approach uses a a 1-bit PPA but makes use the of the SIMON linear approximation and the probing of specific non-linear terms and of the SPECK carry chain techniques as used in the DGPA. Higher order PPA attacks are considered when attacking $\mathbf{Y}$ bits for SIMON d=2,3,4. The SIMON hybrid approach sees a halving of the partitions required for the DGPA $\delta = 2$ as there is a single κ_A term rather than the coupled κ_A, κ_B. This results in extra passes being required for $d = 2, 3, 4$ resulting in more opportunities for a propagation of error from incorrectly recovered key bits. However, for $d \geq 6$ the number of passes is the same as DGPA resulting in an overall reduction of the number of events to carry out the attack at the same level of statistical significance by a factor of $\frac{1}{\sqrt{2}}$ in the zero noise limit. Even with this reduction the recommended $d \geq 9$ for SIMON 64/128 still holds. For SPECK, the number of attacks required to extract key carry chain information, A_C, and the brute force effort required to make use of this information resulting in a minimum effective key strength (MES) is given. The hybrid approach for SPECK sees a decreased number of partitions when attacking the $\mathbf{X}$ register bits, although the number of attacks required to build the chains is substantially increased and the brute force effort increases by a factor of 2^8 for $d = 3, 4, 5$. Even with the reduced partition requirement at $d = 5$, the data requirement still exceeds 2^{32}.

14 Conclusion

The Degenerate Group Power Analysis (DGPA) was shown to be a special case of the all or nothing (AON) Partitioned Power Analysis (PPA) which is evaluated as a t-test allowing for the results to be sorted based on key bit recovery confidence thus minimizing the effort required to brute force the correct key from a mostly correct recovered key. Specific advantage is taken of biases in the degenerate groups relating to the equation structure of both SIMON and SPECK to enhance the recovered key results. Three applications of the DGPA approach were shown. The inverted hierarchy shows minor improvement over other 2-bit PPA approaches for coupled SIMON $\mathbf{X}, \mathbf{Y}^2$ equations at $d = 5$. The identification of key hypothesis forming groups that are degenerate to the correct key G_2 group provides a method to attack SPECK with limited data without having to categorize data samples for all possible key hypotheses. The use of overly defined equation pairs identifies particular key bits which are definitely not recovered in fluctuations of the degenerate G_1 group thereby optimizing brute force attacks on partially recovered keys.

The approaches demonstrated included variants of typical 2-bit AON attacks. All methods used a single power consumption measurement per clock cycle, thus minimizing the amount of data processing required when compared with traditional methods of DPA, CPA, and PPA which examine the entire power trace. It was shown that with minor knowledge of the algorithm being targeted, a specific clock cycle can be attacked further eliminating the possibility for errors that appear in other methods such as DPA ghost peaks which are false signals appearing in the trace in areas where the calculation of interest is not being performed. The attack is found to recover key bits on the PIC16F in the cycles corresponding to the loading of the W register and the output of the ALU value on the data bus, on the MSP430 during register overwrites, and on the Spartan 6 during register overwrites and during changing LUT states. For all attacks attempted, the success is general and not dependent on the algorithm being attacked or the particular instruction being executed.

As the microcontrollers can only execute a single instruction per clock cycle, mitigation techniques, which were not addressed in this paper, must be of the type which limit the number of clocks on which a signal may appear and minimize the effect of any such signal. The Spartan 6 was shown to minimize the usefulness of a leaked signal in an attack by performing d rounds per clock cycle resulting in an increased amount of data and computational resources required to effect full key recovery. Implementations of both SIMON and SPECK 64/128 were experimentally shown to be vulnerable to full key recovery for $d = 1, 2$ on the three platforms examined. On the Spartan, full key recovery of SIMON 64/128 $d \leq 4$ and SPECK 64/128 $d \leq 3$ is achieved in seconds with no more than one million random plaintexts, supporting the use of larger d for most implementations. As a precaution, all such implementations should compute at least 5 rounds for SIMON and 4 rounds for SPECK between register updates to prevent a trivial full key recovery when encrypting less than five million random plaintexts, although a partial key recovery could be achieved with additional effort. Other implementations may have different requirements. It is seen that attacks on SIMON 64/128 $d \geq 9$ and SPECK 64/128 $d \geq 5$ are not feasible, requiring either more than 80-bits of effort or more than 2^n encryptions to be performed with a particular key on the $2n$-bit block cipher. An examination of the reduced data requirement of a 1-bit PPA in the case of zero noise on the power signal finds these limits on d to still require more data than would be available from the use of a given key. The attack on SIMON 64/128 is expected to be computationally unfeasible, defined as requiring more than the available resources or taking more than one year of compute time, at $d = 10$ for a single core machine (2GB RAM, 1TB Storage, 1GFlop/s) and $d = 14$ for a super computer (1PM RAM, 1 EB Storage, 50 PFlop/s), while SPECK 64/128 becomes unfeasible at $d = 5$ for a single core and $d = 6$ for a super computer.

The demonstrated DGPA attack made use of the quantized power consumption of a single register changing state which differed from the power consumption of the same register element maintaining state. As the attack depends only on the signal from the registers and

not from the round logic, these results can be extended to all FPGAs and ASICs of similar design which store the result of d rounds in $2n$ registers and which have registers that are subject to power signal leakage on a state change. An attack on the registers using the methods described on any of the studied implementations which do not have sufficient sample sizes above the recommended minimums must then target the power consumption of the first level LUT output or other circuitry which was shown to be a less effective attack vector on the Spartan.

The success of an attack on a single power measurement per clock cycle supports the feasibility of new power sidechannel attack hardware which is embedded with the low power deployed device under attack for small d implementations. While not tested in this work, the relative simple processing required to calculate the average group power over a single value per clock cycle suggests that a peak sensing ADC along with a low power microcontroller would contain the necessary data acquisition and data processing capability required to perform an attack and to produce a probable key result for exfiltration. The use of expensive lab quality oscilloscopes and data acquisition hardware, high power computing and data storage and processing facilities, and similar requirements of existing power sidechannel attack methods is not required. Calculations of the complexity of such attacks which store then process data or partially process the data during acquisition supports the feasibility of these methods being implemented on memory and computationally constrained platforms.

References

1. P. C. Kocher, J. Jaffe, and B. Jun, Differential Power Analysis in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, (London, UK, UK), pp. 388–397, Springer-Verlag, 1999.
2. E. Brier, C. Clavier, and F. Olivier, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, ch. Correlation Power Analysis with a Leakage Model, pp. 16–29. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.
3. T.-H. Le, J. Clédière, C. Canovas, B. Robisson, C. Servière, and J.-L. Lacoume, *Cryptographic Hardware and Embedded Systems - CHES 2006: 8th International Workshop, Yokohama, Japan, October 10-13, 2006. Proceedings*, ch. A Proposition for Correlation Power Analysis Enhancement, pp. 174–186. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.
4. T. S. Messerges, E. A. Dabbish, and R. H. Sloan, Examining smart-card security under the threat of power analysis attacks *IEEE Transactions on Computers*, vol. 51, pp. 541–552, May 2002.
5. S. Bhasin, S. Guilley, L. Sauvage, and J.-L. Danger, *Topics in Cryptology - CT-RSA 2010: The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings*, ch. Unrolling Cryptographic Circuits: A Simple Countermeasure Against Side-Channel Attacks, pp. 195–207. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.
6. B. Gierlichs, K. Lemke-Rust, and C. Paar, *Cryptographic Hardware and Embedded Systems - CHES 2006: 8th International Workshop, Yokohama, Japan, October 10-13, 2006. Proceedings*, ch. Templates vs. Stochastic Methods, pp. 15–29. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.
7. D. Agrawal, J. R. Rao, P. Rohatgi, and K. Schramm, *Cryptographic Hardware and Embedded Systems – CHES 2005: 7th International Workshop, Edinburgh, UK, August 29 – September 1, 2005. Proceedings*, ch. Templates as Master Keys, pp. 15–29. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.
8. G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi, A testing methodology for sidechannel resistance validation. Cryptography Research Inc. http://csrc.nist.gov/news_events/non-invasive-attack-testing-workshop/papers/08_Goodwill.pdf.
9. T.-H. Le, Q.-T. Nguyen-Vuong, C. Canovas, and J. Cldire, Novel Approaches for Improving the Power Consumption Models in Correlation Analysis Cryptology ePrint Archive, Report 2007/306, 2007. http://eprint.iacr.org/.
10. T. Kugelstadt, Getting the most out of your instrumentation amplifier design Analog Applications Journal, 2005. http://www.ti.com/lit/an/slyt226/slyt226.pdf.
11. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, The SIMON and SPECK Families of Lightweight Block Ciphers Cryptology ePrint Archive, Report 2013/404, 2013. http://eprint.iacr.org/.
12. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, The SIMON and SPECK Block Ciphers on AVR 8-bit Microcontrollers Cryptology ePrint Archive, Report 2014/947, 2014. http://eprint.iacr.org/.