Universiteit Leiden
The Netherlands

Opleiding Informatica

Differences in Side-Channel Power Analysis attacks of selected cryptographic algorithms on different implementation targets

Lisa Schouten

Supervisors:
Nusa Zidaric & Abolfazl Sajadi

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

19/08/2026

Abstract

This thesis investigates differences in Correlation Power Analysis (CPA) attacks on different cryptographic algorithms and implementation targets. CPA attacks were performed on AES-128 and PRESENT-80 across three targets: two ARM microcontrollers using an ARM Cortex-M4 processor core and an FPGA. These are the STM32F303RCT6 and ATSAM4S2AA-MU microcontrollers, and the iCE40 UltraPlus FPGA using a NEORV32 RISC-V soft-core. The attacks were evaluated using the success rate, the number of power traces needed for successful key recovery, and timing statistics.

The results show differences between both the cryptographic algorithms and between the implementation targets. PRESENT required more traces than AES on all three targets to reach a success rate of at least 0.95. The ATSAM4S2AA-MU required more traces than the STM32F303RCT6 for both ciphers, even though the two microcontrollers both use an ARM Cortex-M4 processor core. For the FPGA target, the trigger window start had to be moved so that the targeted operation was included in the stored power trace before full-key recovery was possible. Differences were also observed in the runtime of the attacks.

Overall, the results show that CPA attack performance is influenced by both the cryptographic algorithm and the implementation target. Differences between implementation targets can affect the number of traces and runtime, as well as the attack configuration needed for successful key recovery.

Contents

1 Introduction

Side-channel power analysis attacks exploit variations in the power consumption of a device to recover sensitive information, such as cryptographic keys. The effectiveness of these attacks can depend on different factors, including the cryptographic algorithm and the hardware on which it is implemented.

Previous research has investigated these factors separately. Biryukov et al. [BDG16] compared Correlation Power Analysis (CPA) attacks on several block ciphers implemented on the same target and found differences in the number of traces required for successful key recovery. Other research compared CPA with Differential Power Analysis (DPA) and found advantages for CPA in the number of traces needed or in how clearly the correct key could be distinguished [DL21, LBC17]. CPA has also been successfully used to attack PRESENT in both software and hardware implementations [LBC18, WYW+12]. In addition to differences between algorithms, the implementation target can influence side-channel leakage. Barenghi et al. [BBIP21] found that microarchitectural differences can influence side-channel leakage, even when processors use the same ISA.

This thesis builds on these findings by comparing CPA attacks across both multiple cryptographic algorithms and multiple implementation targets. This allows differences between the evaluated ciphers and differences between hardware implementations to be investigated within the same experimental study. The following research question is investigated: **What differences can be observed in Side-Channel Power Analysis attacks of selected cryptographic algorithms on different implementation targets?**

To investigate this question, CPA attacks are performed on AES-128 and PRESENT-80 across three different implementation targets. Targets 1 and 2 are microcontrollers using an ARM Cortex-M4 core, while target 3 is an FPGA using a NEORV32 RISC-V soft-core. The attacks are evaluated using the success rate, the number of power traces required for successful key recovery, and the attack runtime. The results are compared between the two cryptographic algorithms and between the three implementation targets.

The main contribution of this thesis is the comparison of CPA attacks across different cryptographic algorithms and implementation targets. The comparison includes two different ARM microcontrollers using the same ARM Cortex-M4 processor core, as well as a RISC-V soft-core implemented on an FPGA. The number of traces needed for successful key recovery, success rate, and timing statistics are collected for each target and cipher. These results are then compared and discussed in order to answer the research question.

2 Background

2.1 Cryptography

Cryptography is the science of protecting information for secure communication. Its objective is to create privacy by preventing unwanted parties from reading messages or data.

This is achieved by the encrypting and decrypting of messages and data. Encryption transforms plaintext, which is readable text or data, into ciphertext, which is a version of the data that is not readable without decrypting it first. For decryption, the secret key is needed to turn the ciphertext back into readable plaintext. This cryptographic key is what we want to recover using Side-Channel Analysis (SCA).

Cryptographic algorithms can be divided into symmetric and asymmetric algorithms. Symmetric algorithms use the same key for encryption and decryption. Asymmetric algorithms use two different keys: a public key and a private key. One type of symmetric cipher is the block cipher. The two ciphers we attack in this research are both block ciphers.
Block ciphers encrypt fixed-size blocks of plaintext, usually over multiple rounds. If the plaintext is larger than the block size, it is divided into multiple blocks that are encrypted separately and are often chained in some manner to prevent other attacks. The operations used during the algorithm are designed to provide confusion and diffusion, these are properties of a strong cipher according to Claude Shannon [Sha49].

- **Confusion:** obscures the relationship between key and ciphertext. This is achieved by using nonlinear operations. In AES and PRESENT, this is achieved by substitution operations.
- **Diffusion:** dissipates the statistics of the plaintext across the ciphertext so that a bit of plaintext influences over half of the bits of the ciphertext. The goal of this is to hide statistical properties of the plaintext. In AES, this is achieved with the MixColumns and ShiftRows operations, and in PRESENT with bit permutation in the P-layer. To reach full diffusion in AES, two full rounds are needed; for this reason, an attack on the first-round S-box is possible.

Confusion and diffusion on their own do not make a cipher secure; however, by combining the two operations over multiple rounds, a strong cipher can be created.

2.1.1 AES

The Advanced Encryption Standard (AES) [DBN+01], a symmetric block cipher, is a standardized version of Rijndael [DR99]. It is a substitution–permutation network (SP-network), meaning it is composed of a number of stages each involving substitutions and permutations. It has a block size of 128 bits and supports key sizes of 128, 192, and 256 bits. The number of rounds used depends on the key size: AES-128 uses 10 rounds, AES-192 12 rounds, and AES-256 14 rounds. The cipher operates on a state of 16 bytes that can be represented as a 4×4 matrix. The key can be represented as a matrix with a height of 4 bytes and a width of 4 bytes (AES-128), 6 bytes (AES-192), or 8 bytes (AES-256). The structure of the encryption can be seen in Figure 1. During encryption, the state will go through a number of rounds in which the following operations will be performed: SubBytes, ShiftRows, MixColumns, and AddRoundKey. In the last round, only SubBytes, ShiftRows, and AddRoundKey will be performed, without the MixColumns operation. Decryption works the same as encryption, only all operations are reversed, and the rounds and order of operations are done in reverse.

- **SubBytes:** Each individual byte of the state gets substituted by another byte using a substitution table: the S-Box as can be seen as a lookup table in table 1. The S-Box is

nonlinear, meaning that S-Box(A) + S-Box(B) $\neq$ S-Box(A+B). Because of the nonlinearity, this operation provides confusion.

- **ShiftRows:** The bytes of the state shift cyclically to the left by a row-dependent offset. Row 0 is not shifted, row 1 is shifted by 1, row 2 is shifted by 2, and row 3 is shifted by 3. This spreads the bytes between columns, contributing to diffusion.
- **MixColumns:** Each column in the state is multiplied by a fixed matrix. The result of this is that the bytes of the column are mixed together to produce a new column. This mixing of bytes between columns of ShiftRows combined with mixing of bytes within columns by ShiftRows provides diffusion.
- **AddRoundKey:** The state is XORed with a round key derived from the key using the key-schedule. The XOR (exclusive OR) operation is equivalent to addition modulo 2, so it outputs 1 if exactly one of the input bits is 1 and the other is 0. This operation is applied to each bit of the state and corresponding bit of the round key.

Key Schedule: Expands the key into 11 round keys of 128 bits. The first round key, which is the master key, is added before round 1 of the encryption. The other round keys are added at the end of each round of the encryption. The round keys are computed by a series of operations including rotation, substitution, and XOR. These round keys are then used in the AddRoundKey operation, as shown in Figure 1.

		x															
		00	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**0A**	**0B**	**0C**	**0D**	**0E**	**0F**
y	**00**	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	10	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	20	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	30	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	40	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	50	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	60	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	70	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	80	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	90	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A0	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B0	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C0	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D0	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E0	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F0	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Table 1: AES S-Box, where y represents the 4 most significant bits and x represents the 4 least significant bits of the input byte (in hexadecimal).

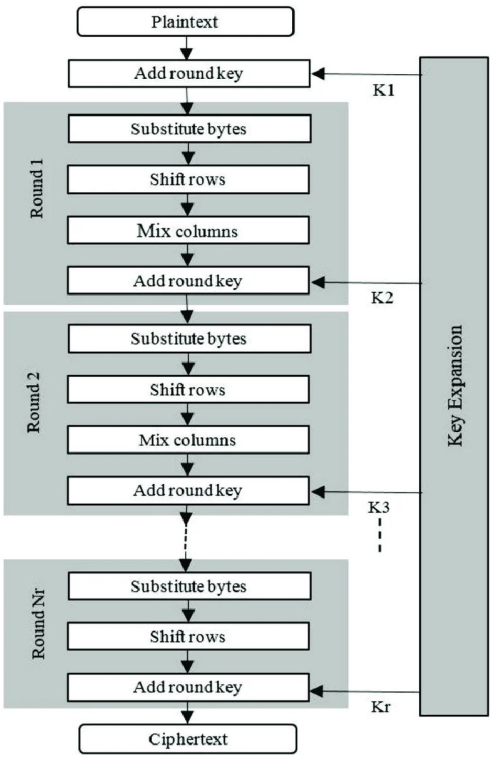

Figure 1: General overview of the AES encryption, taken from [JTS+20].

2.1.2 PRESENT

PRESENT is a block cipher designed with area and power constraints in mind [BKL+07]. The cipher consists of 31 rounds, has a block size of 64 bits and uses a key size of 80 and 128 bits. It is also an SP-network like AES. In each of the 31 rounds, the state is XORed with the round key, passed through the sBoxLayer, and bits are permuted by the pLayer. After the 31 rounds, there is one final XOR with the round key. The structure of the encryption can be seen in Figure 2.

- **addRoundKey:** XORs the state with the round key. The round key is derived from the key schedule.
- **sBoxLayer:** A 4-bit S-box is applied 16 times across the entire 64-bit state (operating on 4-bit nibbles). The S-box can be seen in table 2. The S-Box is nonlinear, because of this, the operation provides confusion.
- **pLayer:** Each bit of the 64-bit state is permuted according to a predefined permutation table (P-table), which is shown in Table 3. The permutation of bits provides diffusion.

Key schedule: Generates 32 round keys of 64 bits: 31 for each round and one additional key for after the final round. It starts with the master key and then feeds the resulting round key back in to generate the next round key, as shown in Figure 2
The first round uses the 64 leftmost bits of the master key. Therefore, it exposes only the leftmost 64 bits of the 80-bit master key.

For each subsequent round after that, the round key will be updated as follows: the 80-bit key will be rotated cyclically to the left by 61 bits. The 4 leftmost bits (bits 79–76) are passed through the S-box The round counter is XORed with bits 19–15. The 64 leftmost bits that result from these three steps are the current round key.
The schedule is invertible, meaning that the original master key can be recovered from the generated round keys by doing the operation in reverse.

x	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
S[x]	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Table 2: PRESENT S-Box, where x is the 4-bit input nibble and S[x] is the 4-bit output nibble (in hexadecimal).

i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
P[i]	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
P[i]	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
P[i]	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
P[i]	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Table 3: PRESENT P-table, where i is the input bit at index and p[i] is the index to which bit i permutes to.

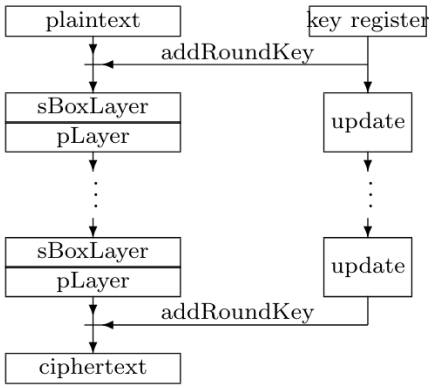

Figure 2: General overview of the PRESENT encryption, taken from [BKL+07].

2.2 Side-Channel Analysis

Side-channel analysis (SCA) exploits unintended side channels that leak information to recover sensitive information, such as secret keys used for cryptographic algorithms. Examples of side channels are power consumption, acoustic emissions, cache, electromagnetic radiation, and timing information. Side-channel power analysis specifically exploits variations in the power consumption of a device to recover sensitive data.

Two side-channel power analysis attacks are Correlation Power Analysis (CPA) and Differential Power Analysis (DPA). These both use power traces that are captured while the target device performs operations.
Each of these traces contains measurements of the device's power consumption during the cryptographic operation. Correlation Power Analysis (CPA) [BCO04] uses the correlation between predicted (hypothetical) values of the algorithm and the measured power traces to recover secret keys. Differential Power Analysis (DPA) [KJJ99] works by comparing the difference between two sets of power traces that are split based on a bit of a predicted intermediate value, such as the output of an S-box for a given key guess. A significant difference between the mean power traces of both sets indicates that the key guess is likely correct.

In this research, CPA is used, which works as follows. CPA calculates the correlation between measured power traces and hypothetical power leakage. Two common leakage models used for this are the Hamming Weight (HW) and Hamming Distance (HD) models. The HW model estimates the power consumption based on the number of bits set to 1 in the resulting state, while the HD model estimates the power consumption based on how many bits change between two states. For the experiments in this research, the HW model is used as the leakage model for all targets. This was chosen because it allows for a consistent comparison of CPA attack performance between the different implementation targets.
The HW assumes that power consumption is correlated with the number of bits that are 1 in the state. Data with more bits set to 1 are expected to consume more power than values with fewer bits set to 1. For example, the number 26 in binary is represented as 00011010. There are 3 bits set to 1, so the HW of 26 would be 3. Another number with a different value can have the same HW, for example, the number 193 (11000001 in binary) also has a HW of 3. Despite the fact that different values can have the same HW, CPA still works because for a correct guess, the HW values for all inputs would correlate with the captured power traces. In contrast, for an incorrect key guess, not all calculated values for all inputs would correlate with the captured power traces.
Using the HW model, hypothetical power consumption values are calculated for each possible value of a portion (ex: a single byte) of the key. The correlation between these predictions and the measured power traces is then computed using the Pearson correlation coefficient.

Pearson correlation coefficient:

$$\rho = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}$$

covariance:

$$\text{cov}(X, Y) = \sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})$$

standard deviation:

$$\sigma_X = \sqrt{\sum_{i=1}^{n} (X_i - \bar{X})^2}$$

Where X and Y are sets of data, n is the number of values, and $\bar{X}$ is the mean of X. Pearson's correlation coefficient measures how much two sets of data are related. It can have a value between -1 and 1, where a value of 1 or -1 means that there is perfect linear correlation, and a value of 0 means there is no correlation between the data sets. Covariance measures whether two sets of data vary together. Standard deviation measures how much the values in a data set differ, on average, from the mean.
In CPA, the Pearson correlation coefficient measures the correlation between the predicted power values and the measured power values. Here, X contains the predicted HW values for a key candidate for each of the N traces, Y contains the measured power values at one sample position across the same N traces, and n is the number of traces N. This calculation is repeated for every sample position in the power trace. For each key candidate, the highest absolute correlation over all sample positions is used as its correlation value. The key candidate with the highest of these values is considered the most likely correct value for that portion of the key. This can then be repeated for all parts of the key to recover the full key.

Common metrics used for SCA include the number of traces, signal-to-noise ratio, attack score (Pearson's correlation coefficient for CPA), guessing entropy, success rate, and correlation [PGA+23]. The metrics looked at in the thesis are the number of traces and success rate. These metrics were selected because they directly evaluate the efficiency and effectiveness of the CPA attacks by measuring the amount of data required and the reliability of key recovery. They also allow for a direct comparison between implementation targets. Other metrics, were not included in order to keep the scope of the research limited.
The **number of traces** is the most common metric used in side-channel analysis, it measures how many power traces need to be collected for a successful attack. A lower number of required traces indicates that the side-channel leakage can be exploited more efficiently, suggesting that the target is more vulnerable to the evaluated attack.
Success rate indicates how reliably the correct key can be recovered over multiple attack attempts. It is the number of times the correct key was found divided by the total number of attacks. A success rate of 1 indicates the correct key was found in all attacks, while a success rate of 0 indicates it was not found in any iteration. A success rate of 0.5 means that in 50% of the attacks the correct key was recovered.

2.2.1 ChipWhisperer

ChipWhisperer is an open-source hardware and software platform designed for side-channel analysis on embedded devices [New25g]. A ChipWhisperer acquisition board controls the target device and is able to capture the power consumption of the target device while the target is performing encryption and convert this to digital data. These power traces can be used for power analysis.

In this research, the CW1173 ChipWhisperer-Lite acquisition board [New25a] is used. The board comes with a built-in ARM target, which is one of the implementation targets that is attacked in the experiments. For attacking the other targets the ChipWhisperer CW313 20-Pin to Card Edge Breakout/Adapter Board [New25e] is used. This adapter allows other target boards to connect to the ChipWhisperer-Lite.

The power traces are captured by ChipWhisperer by measuring changes in the target's current consumption using a shunt resistor placed in the target's power-supply path. Changes in current through the shunt resistor produce corresponding changes in voltage across the resistor. These voltage changes are measured by the ChipWhisperer ADC and from the captured power trace.

2.2.2 Different Targets

The experiments were performed on three different implementation targets: two ARM microcontrollers and one RISC-V soft-core implemented on an FPGA. Target 1 is the built-in CW303 ARM Target containing an STM32F303RCT6 microcontroller [New25b] on the ChipWhisperer-Lite board. Target 2 is the CW312T-SAM4S [New25d] target board containing an Atmel ATSAM4S2AA-MU microcontroller. Target 3 is the CW312T-iCE40 [New25c] target board containing an iCE40 UltraPlus (UP5K) FPGA. The FPGA is configured with a NEORV32 RISC-V soft-core [Nol20]. Targets 2 and 3 are connected to the ChipWhisperer-Lite through the CW313 20-pin adapter.
The setup can be seen in Figure 3 and Figure 4, hardware specifics can be seen in Table 4.

	target 1 **STM32F303RCT6** [STM18]	**target 2** **Atmel ATSAM4S2AA-MU** [Atm15, New25d]	**target 3** **iCE40 UltraPlus (UP5K)** [Lat25, New25c]
Platform	microcontroller	microcontroller	FPGA
Processor	ARM Cortex-M4	ARM Cortex-M4	NEORV32 RISC-V soft-core
Max Clock Frequency	72 MHz	120 MHz	-
Clock Frequency Used	7.38 MHz	7.38 MHz	7.38 MHz
FLASH	256 KB	128 KB	-
SRAM	40 KB	64 KB	-
Vcc	3.3V	1.2V	1.2V
Shunt Resistor	12 Ω	10 Ω	10 Ω
ISA	ARMv7E-M	ARMv7E-M	rv32im_zicsr
IMEM	-	-	64KB
DMEM	-	-	64KB

Table 4: Hardware specifications of the three targets

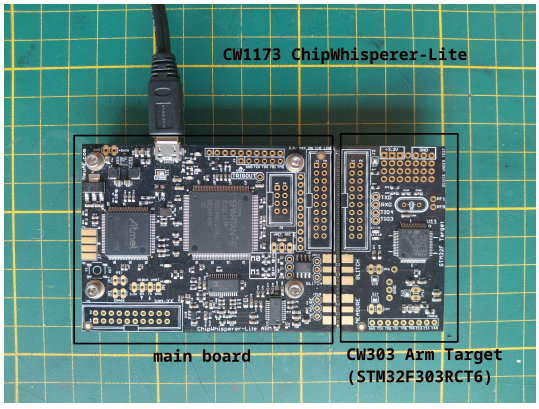

Figure 3: CW1173 ChipWhisperer-Lite with target 1 (CW303 ARM Target)

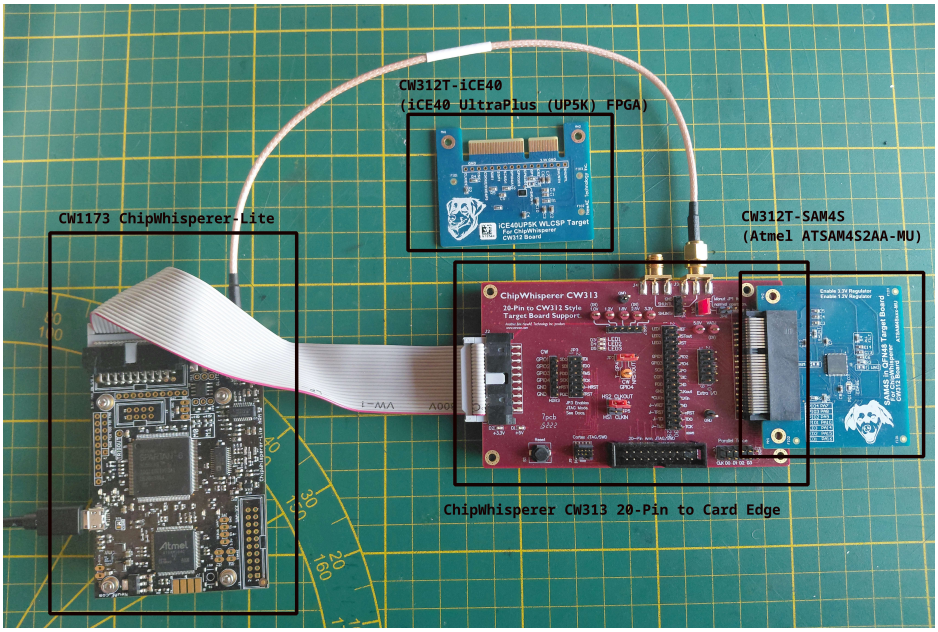

Figure 4: CW1173 ChipWhisperer-Lite setup with the CW313 20-Pin to Card Edge adapter and targets 2 (CW312T-SAM4S) and 3 (CW312T-iCE40)

The clock frequency used during the experiments was 7.38 MHz for all three targets, even though the two microcontrollers support different maximum clock frequencies.
The shunt resistors and supply voltage (VCC) used for the different targets are not the same, which means that the electrical measurement conditions are not identical between all targets.
The Instruction Set Architecture (ISA) specifies the instructions that can be executed by the processor.
The instruction memory (IMEM) is used for storing the program instructions, and the data memory (DMEM) stores data used during program execution. These are used by the NEORV32 soft-core.
The pre-built NEORV32 soft-core used for target 3 was distributed with the ChipWhisperer software [New25f]. The exact LUTs and flip-flop (FF) utilization for this soft-core is not known.

Although both target 1 and target 2 use an ARM Cortex-M4 processor, they are different microcontroller implementations. The Cortex-M4 specifies the processor core architecture and instruction-set architecture, but not the complete microcontroller. The STM32F303RCT6 and ATSAM4S2AA-MU differ in their peripheral systems, memory subsystem, power-management circuitry, clock circuitry, and overall chip implementation. Previous research by Barenghi et al. [BBIP21] has shown that microarchitectural characteristics of a processor can influence side-channel leakage. Comparing these two targets therefore allows differences between microcontroller implementations to be investigated while keeping the processor architecture and instruction-set architecture the same.

Target 3 differs from the first two targets because the NEORV32 processor is not a fixed hardware implementation in silicon. Instead, NEORV32 is a RISC-V soft-core implemented using programmable logic of the iCE40 UltraPlus FPGA. The power consumption of the target does not only reflect the processor operation but also the underlying FPGA implementations. As a result, the FPGA target could exhibit differences in leakage characteristics, execution timing, and noise

characteristics compared with the two microcontroller targets.

3 Related Work

Previous research by Biryukov et al. [BDG16] evaluated the resistance of several lightweight block ciphers implemented on an 8-bit AVR processor against CPA. Their research showed that the AES implementation that had no side-channel countermeasures required the fewest power traces for successful key recovery compared to the other evaluated ciphers. After that, the second lowest number of traces was required by the ciphers that used 4-bit S-boxes (LBlock, Piccolo, and PRINCE). PRESENT was not included in their comparison but also uses a 4-bit S-box. This work showed that CPA effectiveness can differ based on the cipher used.

Other research has compared DPA and CPA on AES. Di Lorenzo [DL21] compared DPA and CPA on AES using a ChipWhisperer NANO. The attacks used a HW leakage model. The results showed that CPA was able to recover AES keys using approximately 3 times fewer traces compared to DPA. It also showed that using the SubBytes operation as an attack target needed fewer traces compared to using the AddRoundKey operation as the target.
Another paper looked at the difference between DPA and CPA on AES implemented on an Arduino Uno [LBC17]. Their results showed that both DPA and CPA were able to recover the full correct AES key. However, the CPA results were easier to interpret because they contained less noise than the DPA results, making the correct key bytes more distinguishable.
Both papers showed that, in their experiments, CPA had an advantage over DPA in terms of the number of traces needed or the distinguishability of the correct key bytes. For this reason CPA is used in this thesis.

Sajadi et al. [SZSM24] compared different countermeasures against SCA attacks on Tiny-AES implemented on an Ibex RISC-V core on an FPGA. The countermeasures looked at included hardware and software noise generation, dummy instruction insertion, and software masking. These countermeasures aim to protect against power side-channel attacks. Noise generation adds unrelated noise to the power signal, this makes useful leakage harder to distinguish. Dummy instructions are inserted during operations to change the execution timing, which makes the attack harder. Software masking hides sensitive intermediate values by using random masks, which makes it harder to extract sensitive information from power traces. Their results showed that the noise generation countermeasures could be bypassed using exhaustive search or by increasing the number of power traces. Their results also showed that adding dummy instructions and software masking provided greater protection but introduced additional overhead.

Other previous research performed CPA on a software implementation of the PRESENT cipher running on an Arduino Uno [LBC18]. Two different CPA attack experiments were performed, one targeting the addRoundKey operation of the first round of the cipher, and the other experiment targeting the S-box operations of the first round. Both experiments used the Hamming Weight leakage model. Their results showed that the S-box operation was a better target than the addRoundKey operation. With the S-box as the target, the first 8 of the 10 bytes of the key could be recovered. The remaining 2 bytes of the key were then recovered using brute force by running

the cipher with the found key and all possible values for the 2 unknown bytes until the resulting ciphertext matched the known ciphertext.
CPA has also been performed on a simulated hardware implementation of PRESENT [WYW+12]. Wang et al. proposed an attack method that does not require resetting the target device before each measurement. They compared a 4-bit model to an 8-bit model attack on PRESENT. Their 4-bit model uses the Hamming Distance between two successive 4-bit states, while their 8-bit model uses the Hamming Distance between two successive 8-bit states. Their results showed that their 8-bit model needed at least 230 plaintexts and their 4-bit model needed 450 plaintexts to successfully recover the first-round key. However, their 8-bit attack required considerably more time than their 4-bit attack. Since their attacks used a simulated hardware implementation, the results are less relevant to this work because this thesis attacks software implementations of the ciphers and uses physical power traces.
These works show that both software and hardware implementations of PRESENT are vulnerable to CPA attacks.

Barenghi et al. [BBIP21] proposed a framework for characterizing microarchitectural side-channel leakage at the Instruction Set Architecture (ISA) level. They evaluated the framework on two ARM processors, a Cortex-M4 (STM32F401) and a Cortex-M7 (STM32F746ZG), using electromagnetic side-channel measurements. The two processors implement the same ARMv7-M Thumb ISA but have different microarchitectures. The Cortex-M4 is a single-issue processor, while the Cortex-M7 is a dual-issue processor with a different internal processor structure. Barenghi et al. found differences in the leakage from internal processor components. Leakage from the register file and an internal pipeline register was observed on both processors. Leakage from an additional EX/WB pipeline register was observed on the Cortex-M7 but not on the Cortex-M4. Leakage associated with the load-store unit was also observed on both processors. They validated that the identified leakage could be exploited by performing CPA attacks on unprotected software AES implementations on both processors. Their work showed that microarchitectural differences can influence side-channel leakage, even when processors use the same ISA.

Previous work has shown the vulnerability of PRESENT against power analysis attacks, looked at the differences in CPA and DPA attacks, evaluated countermeasures against SCA, and investigated the influence of processor microarchitecture on side-channel leakage. This thesis aims to build on these results by comparing CPA attacks on AES and PRESENT on multiple implementation targets.

4 Experimental Setup

The CPA attacks were performed on the AES-128 and PRESENT-80 ciphers across three implementation targets. The resulting success rate, number of traces, and attack times were measured and compared. The three implementation targets are the STM32F303RCT6 microcontroller, the Atmel ATSAM4S2AA-MU microcontroller, and the iCE40 UltraPlus FPGA with a NEORV32 RISC-V soft-core. These targets are introduced in more detail in the Different Targets section (2.2.2) in the Background.

Experiments on target 1 were performed using ChipWhisperer software version 5.6.1. Experiments on targets 2 and 3 were performed using ChipWhisperer software version 6.0.0. Target 1 was tested first, and ChipWhisperer was updated before testing targets 2 and 3 because the setup for target 2 did not work correctly with version 5.6.1.

The compiler used for targets 1 and 2 was `arm-none-eabi-gcc` version 8.3.1 with the `-Os` optimization setting, which optimizes for output binary size. The `-Os` flag also enables the optimizations from the `-O2` flag, which lets GCC perform nearly all supported optimizations that do not increase the size [Fre].
Target 3 used the `xpack-riscv-none-elf-gcc-15.2.0-1` compiler with the `-Os` optimization setting.

The CPA calculations for the experiments were performed in a Debian virtual machine using 2 cores and 2 GB of memory. The virtual machine was run on an HP ZBook Firefly 15 inch G8 Mobile Workstation PC with an 11th Gen Intel® Core™ i7-1165G7 processor running at 2.80GHz. All time statistics were measured in this environment. The hardware and targets used are explained in the ChipWhisperer (2.2.1) and Different Targets (2.2.2) sections.

4.1 Trace Acquisition

The acquisition settings that were used during the capturing with the ChipWhisperer can be seen in Table 5. The parameters shown in the table are the same for all targets and ciphers.

Gain determines how much the measured signal is amplified before it is sampled by the Analog-to-Digital Converter (ADC). The gain mode determines the amplification range used by the ChipWhisperer. The ADC converts the measured analog voltage signal into digital samples that make up the power trace. The trigger source determines when the acquisition starts, a rising edge means that acquisition is triggered when the trigger signal changes from low to high. The samples per trace determine how many ADC samples are stored for each captured power trace. The ADC clock source is the clock used for sampling by the ADC, and the ADC rate determines how many samples are taken per second. The samples per target clock cycle are the number of samples that are taken in a target clock cycle.

	Acquisition parameters
Gain mode	high
Gain	24.8 dB
Trigger source	rising edge
Samples per trace	5000
ADC clock source	clkgen_x4
ADC rate	29.54 MHz
Clock generator frequency	7.38 MHz
Samples per target clock cycle	4.00

Table 5: Acquisition parameters for all targets

Table 6 shows the trigger window used for the different ciphers and targets. The trigger window indicates which part of the encryption was marked by the trigger signal in the target code, with the trigger set high at the start of the window and low at the end. The trigger interval shows the start

and end of the trigger window in target clock cycles. The stored interval shows which target clock cycles are included in the 5000 stored ADC samples. The full round interval shows the measured start and end of the attacked round in target clock cycles. The percentage of the full round stored is calculated from the overlap between the stored interval and the full round interval. The start of the trigger also starts the acquisition, but setting the trigger low does not stop the acquisition. As shown in Table 5, the number of samples per trace was 5000. The acquisition continues until all 5000 samples have been stored. At 4.00 ADC samples per target clock cycle, this corresponds to approximately 1250 target clock cycles being stored in each trace. Therefore, the trigger window and the interval stored in the power trace are separate values.

		Trigger window	**Trigger interval (target cycles)**	**Stored interval (target cycles)**	**Full round interval (target cycles)**	**Percentage of full round stored**
AES target 1	R1	full encryption	0 - 6952	0 - 1250	308 - 994	100%
AES target 2	R1	full encryption	0 - 8088	0 - 1250	279 - 1266	98.4%
AES target 3	R1	round 1	1110 - 3277	1110 - 2360	1110 - 3772	47.0%
PRESENT Target 1	R1	full encryption	0 - 53750	0 - 1250	920 - 3536	12.6%
	R2	round 2 (S-Box)	2707 - 2894	2707 - 3957	2621 - 4314	73.8%
PRESENT target 2	R1	full encryption	0 - 54058	0 - 1250	1066 - 2786	10.7%
	R2	round 2 (S-Box)	2846 - 3106	2846 - 4096	2791 - 4512	72.6%
PRESENT target 3	R1	round 1 (S-Box)	12888 - 13867	12888 - 14138	12331 - 24252	10.5%
	R2	round 2 (S-Box)	24816 - 25804	24816 - 26066	24270 - 36180	10.5%

Table 6: Trigger intervals, stored intervals, and full-round intervals for all ciphers on all targets, where R1 is the attack on round one and R2 is the attack on round two.

The percentages in Table 6 are approximate. Additional trigger points were inserted into the target code to determine the locations of the rounds. These additional trigger operations introduced a small amount of additional execution overhead, so the measured round intervals can differ slightly from the execution timing of the firmware used in the experiments.

Before reducing the trigger window for target 3, a full-encryption trigger window was tested for both AES and PRESENT. For AES, this full-encryption trigger window lasted 85940 ADC clock cycles, corresponding to 21485 target clock cycles. For PRESENT, it lasted 1529624 ADC clock cycles, corresponding to 382406 target clock cycles. Both configurations still stored 5000 samples per trace and used no decimation. These full-encryption configurations were not used for the final experiments because they did not result in successful full-key recovery on target 3.

The results in Table 6 show that the complete attacked round was not always stored in the power trace. For several experiments, only part of the attacked round was stored while successful key recovery was still possible. This is discussed further in the Success Rate paragraph of the Discussion.

The plaintexts used for each captured trace were randomly generated. For each iteration of the attack, a new set of power traces was captured and a new random key was generated. The target device was not reset between individual encryption operations during trace capture for either AES or PRESENT.

4.2 CPA on AES-128

The AES implementation used in the experiments was the `tiny-AES128-C` implementation included with ChipWhisperer [New]. The same implementation was used in ChipWhisperer 5.6.1 and 6.0.0. This implementation is derived from the tiny-AES project by Kokke [Kok19]. This variant of AES uses a 128-bit (16-byte) key.

The CPA attack targeted the output of the SubBytes operation in the first round. For each hypothetical key byte key_i, the corresponding plaintext byte pt_i was XORed with the hypothetical key byte, after which the 8-bit S-box was applied to the result of this XOR operation. The predicted intermediate value was calculated as

$$state_i = Sbox[pt_i \oplus key_i]$$

where i is the byte index, which goes from 0 to 15. The HW of each resulting $state_i$ was calculated and used as the hypothetical leakage.

For each of the key bytes, the hypothetical leakage for all 256 possible byte values was calculated. The correlation of each key value with the captured power traces was calculated. The hypothetical key byte with the highest correlation was selected. Finally, the 16 best key bytes were then combined to reconstruct the predicted master key.

For targets 1 and 2, the trigger window covered the full encryption. However, for target 3, no fully correct key was able to be recovered using power traces captured with a trigger window covering the full encryption. The corresponding trigger window sizes can be seen in Table 6. Therefore, the trigger window was reduced to only cover round one, this resulted in full master key recovery being possible.

4.3 CPA on PRESENT-80

The PRESENT implementation running on the target device is based on a C implementation of PRESENT by Petar Tonkovic (Pepton21) [Ton19]. Several changes to this implementation were made. The original implementation worked with `char*` for the key and plaintext, this was changed to work with `uint8_t*`. Header files were added to the implementation. The implementation was also modified to work with Simpleserial (version 1.1) [New26], which was used for communication between the ChipWhisperer capture and target device.

The whole 80-bit master key can not be recovered by attacking a single round of the cipher. Only 64 bits of the master key can be recovered from the first round, since the first-round key only uses the first 64 bits of the master key. The first-round attack leaves the last two bytes of the master key unknown. These missing bytes can be recovered by brute force, as is done in the research by Lo et al. [LBC18], or from the second-round key. In these experiments, the missing bytes were recovered by attacking round two. After the second-round key is recovered, the missing bytes can be obtained by reversing the steps of the key schedule. These bytes can then be combined with the round-one key to reconstruct the full master key. The traces for the round-one and round-two attacks were

captured separately, with the same number of traces captured for each attacked round.

The CPA attack targeted the output of the S-Box operations. This was found to be an effective target by Lo et al. [LBC18]. Each hypothetical round-key nibble k_i was XORed with the input state nibble x_i after which the 4-bit S-box was applied. The predicted output state nibble v_i was calculated as

$$v_i = Sbox[x_i \oplus k_i]$$

Where i is the index from 0 to 15 of the nibble in the round-key and state.
For round one, the input state nibble x_i, is the plaintext nibble pt_i:

$$x_i = pt_i$$

For round two, the input state x of the first formula is the output state of round one of the cipher. This output of round one of the cipher is calculated using the plaintext and round-one key $K1$:

$$x_i = [pLayer(sBoxLayer(pt \oplus K1))]_i$$

Where $K1$ is the correct round-one key and x_i is nibble i of the output state of round one, which is the input state to round two. Without a fully correct round-one key, the correct round-one output state cannot be calculated, and thus no fully correct round two key can be recovered.
The attack on PRESENT uses the HW leakage model, as was used for the attack on AES. For both the attacks on round 1 and 2, the HW of the output state nibble v_i was calculated. Since PRESENT uses a 4-bit S-box, the hypothetical leakage is calculated from the predicted 4-bit S-box output for each key-nibble guess.

For the attack on the first round key on targets 1 and 2, the trigger window covered the full encryption. For target 3, when using a trigger window covering the full encryption, no full master key was able to be recovered. Therefore, the trigger window was reduced to only cover the first round of encryption, after which full-key recovery became possible. A possible explanation for why the full-encryption trigger window did not result in successful key recovery, and why reducing the trigger window improved the attack, is discussed in the Success Rate paragraph of the Discussion. For all three targets, the trigger window for the attack on the second round only covered the second round. The corresponding trigger-window sizes can be seen in Table 6.

4.4 Metrics and Statistics

The metrics used to evaluate the attacks were the number of traces and the success rate. The number of traces (N) is the number of power traces used to perform CPA. For PRESENT, N refers to the number of traces used per attacked round. Since round one and round two were captured separately, a PRESENT attack using N traces per round required $2N$ captured traces in total. The success rate is the rate at which the attack successfully recovered a correct master key.
To allow comparison between the experiments, the lowest tested number of traces that achieved a success rate ≥ 0.95 was used. A success rate ≥ 0.95 was chosen because this allowed for reliable recovery while allowing occasional unsuccessful attacks. Because only specific numbers of traces were tested, the 0.95 threshold was given as a range between the last tested value below 0.95 and

the first value that reached it. For example, $(N_1, N_2]$ means the threshold was reached somewhere between N_1 and N_2 traces, but the exact value is unknown.
Each experiment was repeated 50 times for each number of traces. The uncertainty of the observed success rate was estimated using the standard error.

$$SE = \sqrt{\frac{p(1-p)}{n}}$$

Where p is the observed success rate and n is the number of attack iterations. The standard error is shown as ± 1 SE error bars in the success rate figures.
The plaintexts used for each captured trace were randomly generated. For each iteration of the attack, a new set of power traces was captured. A new random key was used for each iteration.
The following statistics were collected for each experiment: capture time, processing time, and total time. The capture time is the time needed for capturing the specified number of traces. This time includes the execution of the full encryption and is not only the duration of the trigger window. The processing time is the time it takes to perform CPA on all key bytes or nibbles and reconstruct a complete key. The total time is the sum of the capture and processing time.

For each experiment done, the results are presented in the from of a table with the mean value of these metrics and statistics for different numbers of traces. These results will also be visualized in a plot of the average success rate for different numbers of traces with standard error bars, and a plot with the average time statistics for different number of traces.

5 Experiments and Results

The six experiments are organized by cipher and target, as can be seen in Table 7. Each experiment reports the measured metrics and statistics. First, the results of the CPA attack on AES are presented for all targets, followed by a summary comparing the AES results of all three targets. Then, the results for CPA on PRESENT are presented for all targets, followed by a summary comparing the PRESENT results of all three targets.

	Experiments			
	Target 1	**Target 2**	**Target 3**	**Summary**
AES	5.1.1	5.1.2	5.1.3	5.1.4
PRESENT	5.2.1	5.2.2	5.2.3	5.2.4

Table 7: Overview of the experiment for each cipher and target, with summary subsections comparing the results of each cipher across the different targets

5.1 AES

5.1.1 Target 1

CPA was tested on AES-128 on the STM32F303RCT6. The experiments consisted of running CPA with different amounts of power traces: 10, 20, 25, 30, 35, 40 and 50. For each number

of power traces, the CPA attack was repeated 50 times to accurately measure the mean accuracies, capture times, and processing times. The results of this can be found in Figure 5, 6 and Table 8.

Figure 5: Success rate (over 50 iterations) for varying numbers of power traces when attacking AES on Target 1

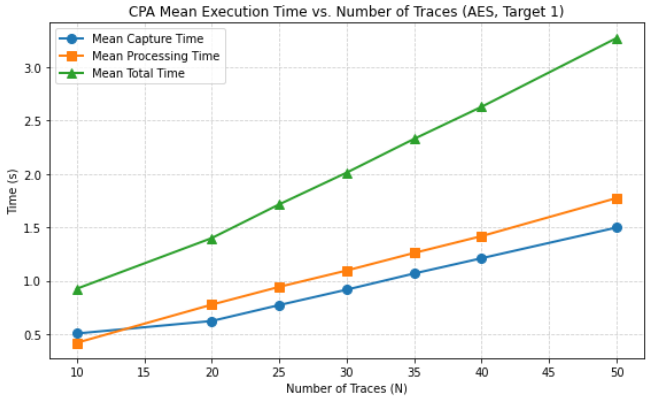

Figure 6: Average attack time (over 50 iterations) for varying numbers of power traces when attacking AES on Target 1

Figure 5 shows the average success rate when running 50 CPA attacks with varying numbers of traces. The success rate is the number of times the correct key was found divided by the total number of attacks, 50 in this case. A success rate of 1 indicates the correct key was found in all 50 iterations, while a success rate of 0 indicates it was not found in any iteration.
The results show that the success rate increases as the number of traces grows, and from 40 traces onward the correct key was recovered consistently with a perfect success rate of 1.00. However, the comparison between experiments uses the predefined success-rate threshold of at least 0.95 rather than requiring a perfect success rate of 1.00. The lowest tested value that passed this threshold was 30 traces, with a success rate of 0.96. This shows that recovering the correct key for AES-128 is feasible using CPA.

Figure 6 shows that increasing the number of traces results in a linear increase in the mean capture time and processing time. For all evaluated trace counts, the difference between the capture and processing time is small, with an average difference of 0.18 seconds.

When taking the results of Figure 5 and Table 8 into account, we can see that the lowest tested number of traces needed to reach a success rate of at least 0.95 is 30. For 30 traces, the success rate is 0.96 with a total mean capture and processing time of 2.02 seconds. The success rate of 0.95 is reached somewhere between 25 traces with a success rate of 0.80 and 30 traces with a success rate of 0.96, so the trace count threshold is bracketed by $(25, 30]$.

Number of Traces (N)	Success Rate	Mean Capture Time (s)	Mean Processing Time (s)	Mean Total Time (s)
10	0.00	0.51	0.42	0.93
20	0.14	0.62	0.78	1.40
25	0.80	0.77	0.94	1.71
30	0.96	0.92	1.10	2.02
35	0.98	1.07	1.26	2.33
40	1.00	1.21	1.42	2.63
50	1.00	1.50	1.77	3.27

Table 8: Results for different numbers of traces (over 50 iterations) on AES on Target 1

5.1.2 Target 2

CPA was performed on the AES-128 algorithm running on the Atmel ATSAM4S2AA-MU microcontroller. The attack was run with 100, 150, 200, 300, 400, and 500 power traces. For each number of power traces, the CPA attack was repeated 50 times. The results of this can be found in Figure 7, 8 and Table 9.

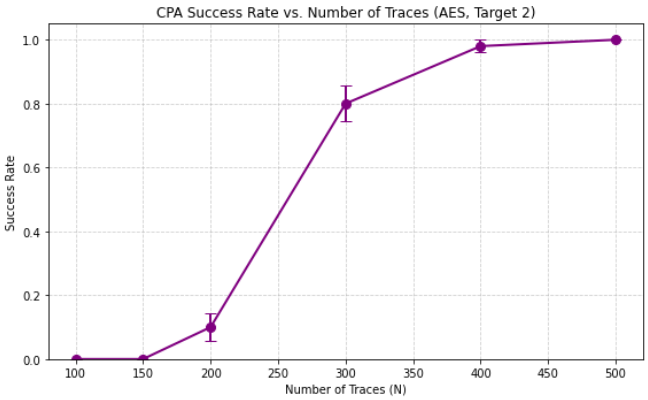

Figure 7: Success rate (over 50 iterations) for varying numbers of power traces when attacking AES on Target 2

Figure 8: Average attack time (over 50 iterations) for varying numbers of power traces when attacking AES on Target 2

Figure 7 shows the average success rate when running 50 CPA attacks with varying numbers of traces. The results show that with enough traces, recovering the correct key for AES-128 is feasible using CPA. The success rate increases with the number of traces. The largest improvement occurred between 200 and 300 traces, where the success rate went from 0.10 to 0.80. The increase in success rate slows down significantly between 400 and 500 traces, where the increase only goes from 0.98 to 1.00. At 500 traces, the success rate reaches a value of 1.00, meaning it was able to recover the key each time.

From Figure 8, it can be seen that increasing the number of traces leads to an approximately linear increase in mean capture and processing time. The mean Processing time is longer than the mean capture time for all numbers of traces.

From Figure 7 and Table 9, it can be seen that the lowest tested number of traces that reached a success rate of at least 0.95 was 400 traces. With 400 traces, the success rate was 0.98, with a mean total capture and processing time of 43.39 seconds. The success rate threshold of 0.95 is reached somewhere between 300 traces with a success rate of 0.80 and 400 traces with a success rate of 0.98, so the trace count threshold is bracketed by $(300, 400]$.

Number of Traces (N)	**Success Rate**	**Mean Capture Time (s)**	**Mean Processing Time (s)**	**Mean Total Time (s)**
100	0.00	3.63	5.85	9.48
150	0.00	4.74	10.18	14.92
200	0.10	7.26	13.97	21.23
300	0.80	10.21	21.26	31.47
400	0.98	13.80	29.59	43.39
500	1.00	20.13	37.63	57.76

Table 9: Results for different number of traces (over 50 iterations) on AES on Target 2

5.1.3 Target 3

CPA was performed on the AES-128 algorithm running on an iCE40 UltraPlus (UP5K) FPGA with a NEORV32 soft-core RISC-V processor.
For this target, no fully correct key could be recovered when using the same trigger-window configuration as for targets 1 and 2. This was tested with 100, 500, 1000, 3000, 5000, and 10000 traces with no fully correct key being recovered. For targets 1 and 2, the trigger window was set around the full encryption of AES. Reducing the size of the trigger window from the full encryption to only round one of encryption made correct key recover possible.

The attack was performed with 10, 15, 20, 25, 30, and 40 power traces. For each number of power traces, the CPA attack was repeated 50 times. The results of the attack with the smaller trigger window can be seen in Figure 9, 10 and Table 10

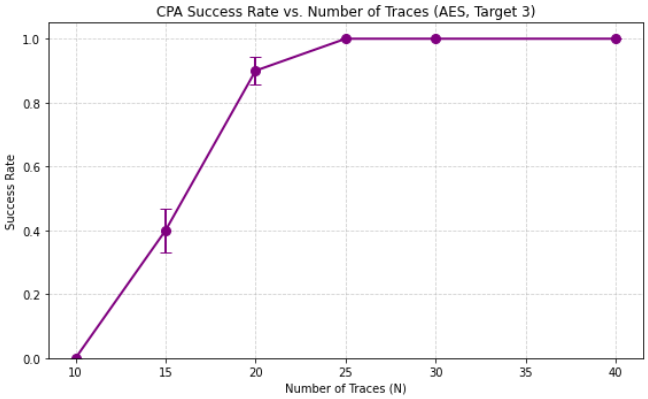

Figure 9: Success rate (over 50 iterations) for varying numbers of power traces when attacking AES on Target 3

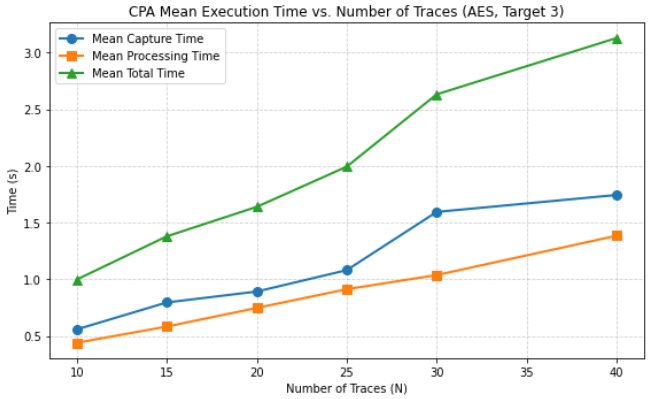

Figure 10: Average attack time (over 50 iterations) for varying numbers of power traces when attacking AES on Target 3

Figure 9 shows the average success rate when running 50 CPA attacks with varying numbers of

traces. The results show that CPA is feasible for AES-128. The success rate increases with the number of traces,with a larger increase at first, which slows down with more traces until it reaches a perfect success rate of 1.00 from 25 traces onward.

Figure 10 shows that the mean capture and mean processing time follow an approximately linear increase with the number of traces. The mean capture and processing time for each number of traces is roughly the same.

From Figure 9 and Table 10, it can be seen that the lowest tested number of traces that reached a success rate of at least 0.95 was 25. With 25 traces, the attack had a perfect success rate of 1.00, with a mean total capture and processing time of 1.99 seconds. The success rate threshold of 0.95 is reached somewhere between 20 traces, with a success rate of 0.90, and 25 traces, with a success rate of 1.00, so the trace count threshold is bracketed by $(20, 25]$. These results were obtained using the reduced trigger window that only covered round one. With the full-encryption trigger window, no correct full key was recovered during testing.

Number of Traces (N)	**Success Rate**	**Mean Capture Time (s)**	**Mean Processing Time (s)**	**Mean Total Time (s)**
10	0.00	0.56	0.44	1.00
15	0.40	0.80	0.58	1.38
20	0.90	0.89	0.75	1.64
25	1.00	1.08	0.91	1.99
30	1.00	1.59	1.04	2.63
40	1.00	1.74	1.39	3.13

Table 10: Results for different number of traces (over 50 iterations) on AES on Target 3

5.1.4 Summary

The results for all targets show that CPA attacks are feasible on AES on the three tested targets. The number of traces needed to achieve a success rate of at least 0.95 was different for the three targets.

Target 3 required the fewest power traces, only needing 25 traces to achieve a success rate of at least 0.95 with a mean total attack time of 1.99 seconds. However, target 3 used a reduced trigger window while the other targets used trigger windows covering the full encryption. Without this smaller trigger window, target 3 was not able to recover any fully correct keys with the tested number of traces. Therefore, the lower number of traces for target 3 cannot be attributed only to differences in implementation.
Target 1 required slightly more traces to reach a success rate of at least 0.95, needing 30 traces to achieve a success rate of 0.96, taking on average 2.02 seconds total.
Finally, target 2 required the most traces to reach a success rate of at least 0.95, needing 400 traces to achieve a success rate of 0.98 taking on average 43.39 seconds total.

Overall, under the tested configurations, target 3 required the fewest traces, followed closely by target 1. Target 2 required the most traces, needing significantly more traces than the other two

targets. However, the ranking of target 3 should be interpreted with the difference in trigger window in mind.

5.2 PRESENT

5.2.1 Target 1

CPA was performed on the PRESENT-80 algorithm running on the STM32F303RCT6 microcontroller with an ARM Cortex-M4 core. To gather the results for the experiments, the CPA attack was repeated 50 times for the following number of traces (N): 50, 100, 150, 200, 300. For each tested value of N, N traces were captured for the attack on round one and another N for the attack on round two. The success rate, capture, and processing time of the attack were measured for both rounds. The results of this can be found in Figure 11, 12 and Table 11.

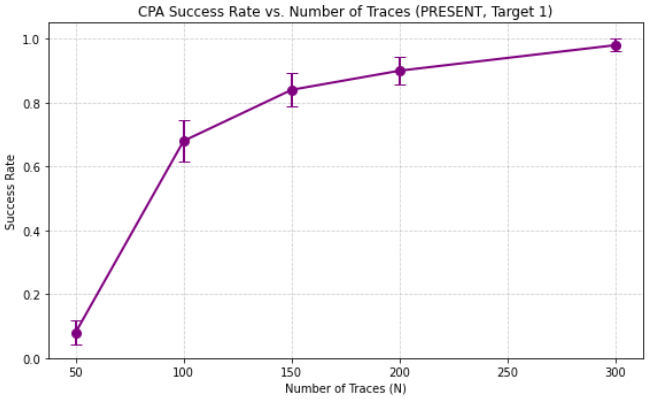

Figure 11: Success rate (over 50 iterations) for varying numbers of power traces per attacked round when attacking PRESENT on Target 1

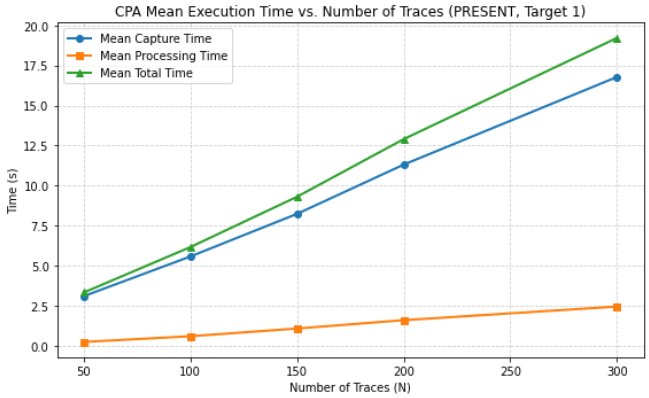

Figure 12: Average attack time of both attacked rounds (over 50 iterations) for varying numbers of power traces per attacked round when attacking PRESENT on Target 1

Figure 11 shows the average success rate when running 50 CPA attacks with varying numbers of traces. At 300 traces, the attack was able to recover the key with a success rate of 0.98.

Figure 12 shows that the average capture and processing time increases roughly linearly with the number of traces. Traces of 2 rounds needed to be captured and processed which increased the total time needed. The mean processing time is noticeably smaller than the mean capture time for all numbers of traces. This could be explained by the fact that the analysis is performed per nibble (4 bits) and not per byte (8 bits). For each nibble, only 16 values need to be evaluated compared to 256 for each byte.

From Figure 11 and Table 11, it can be seen that the lowest tested number of traces that reached a success rate of at least 0.95 was 300. With 300 traces, the success rate was 0.98, with a mean total capture and processing time of 19.23 seconds. The success rate threshold of 0.95 is reached somewhere between 200 traces with a success rate of 0.90 and 300 traces with a success rate of 0.98, so the trace count threshold is bracketed by $(200, 300]$. All reported trace counts for PRESENT are per attacked round.

Number of Traces (N)	Success Rate	Mean Round 1 Time (s)			Mean Round 2 Time (s)			Mean Total Time (s)
		Capture	Processing	Total	Capture	Processing	Total	
50	0.08	1.50	0.11	1.61	1.59	0.12	1.71	3.32
100	0.68	2.70	0.29	2.99	2.86	0.30	3.16	6.15
150	0.84	4.09	0.55	4.64	4.14	0.52	4.66	9.30
200	0.90	5.56	0.80	6.36	5.75	0.79	6.54	12.90
300	0.98	8.22	1.23	9.45	8.56	1.22	9.78	19.23

Table 11: Results for different number of traces (over 50 iterations) on PRESENT on Target 1

5.2.2 Target 2

PRESENT was performed on the PRESENT-80 algorithm running on the Atmel ATSAM4S2AA-MU microcontroller. The experiments consisted of running CPA with the following number of traces (N): 100, 150, 200, 300, 400, 500, 600, and 700. For each tested value of N, Ntraces were captured for the attack on round one and another N for the attack on round two. The success rate, capture, and processing time of the attack were measured for both rounds. The results of this can be found in Figure 13, 14 and Table 12.

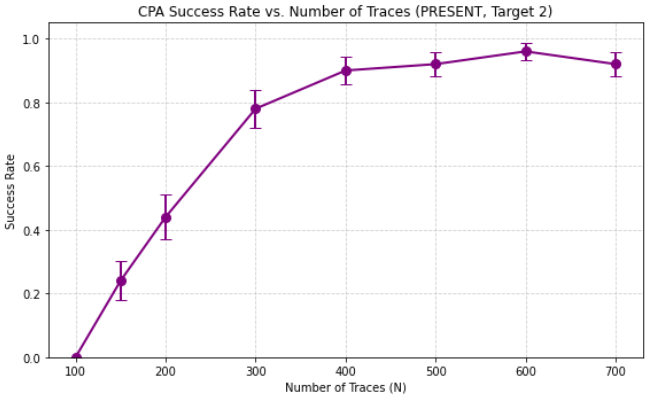

Figure 13: Success rate (over 50 iterations) for varying numbers of power traces per attacked round when attacking PRESENT on Target 2

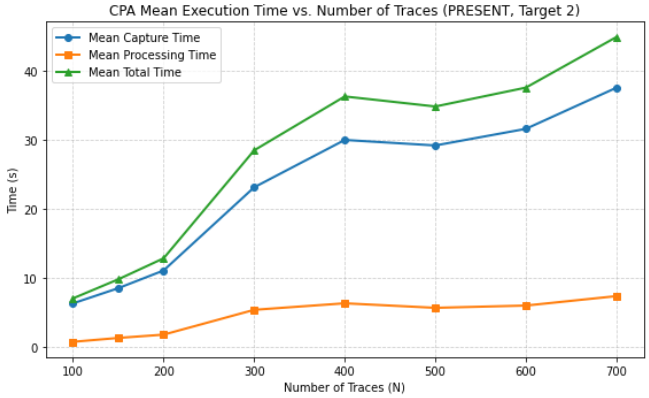

Figure 14: Average attack time of both attacked rounds (over 50 iterations) for varying numbers of power traces per attacked round when attacking PRESENT on Target 2

From Figure 13, it can be seen that at 600 traces, a success rate of 0.96 was achieved with a standard error of 0.028. Both 500 and 700 traces had a success rate of 0.92 with a standard error of 0.038. These differences in success rate should not be interpreted as meaningfully different, because the error bars overlap.

Figure 14 shows that the mean processing time is noticeably smaller than the mean capture time. As explained in the target 1 section of PRESENT 5.2.1, this could be explained by the attack being performed by nibble instead of per byte. From the figure, it can also be seen that the time statistics are less linear than for the other experiments. A possible explanation of this is that it is caused by other processing running on the computer during the capturing and processing of this

experiment. This was not investigated further.

When taking the results of Figure 13 and Table 12 into account, we can see that the lowest tested number of traces needed to reach a success rate of at least 0.95 was 600.For 600 traces, the success rate was 0.96 and the total mean capture and processing time was 37.63 seconds. Based on the tested number of traces, the success rate threshold of 0.95 was reached somewhere between 500 traces with a success rate of 0.92 and 600 traces with a success rate of 0.96, so the threshold is bracketed by $(500, 600]$. However, due to the uncertainty of the estimated success rates, this interval should not be interpreted as an exact estimate for the 0.95 success rate threshold. All reported trace counts for PRESENT are per attacked round.

Number of Traces (N)	**Success Rate**	**Mean Round 1 Time (s)**			**Mean Round 2 Time (s)**			**Mean Total Time (s)**
		Capture	**Processing**	**Total**	**Capture**	**Processing**	**Total**	
100	0.00	3.13	0.36	3.49	3.18	0.36	3.54	7.03
150	0.24	4.21	0.64	4.85	4.30	0.64	4.94	9.79
200	0.44	5.56	0.88	6.44	5.52	0.88	6.40	12.84
300	0.78	11.41	2.68	14.09	11.74	2.69	14.43	28.52
400	0.90	15.51	3.11	18.62	14.51	3.21	17.72	36.34
500	0.92	14.83	2.82	17.65	14.39	2.84	17.23	34.88
600	0.96	15.71	2.99	18.70	15.93	3.00	18.93	37.63
700	0.92	18.48	3.67	22.15	19.13	3.69	22.82	44.97

Table 12: Results for different numbers of traces (over 50 iterations) on PRESENT on Target 2

5.2.3 Target 3

CPA was performed on the PRESENT-80 algorithm running on an iCE40 UltraPlus (UP5K) FPGA with a NEORV32 soft-core RISC-V processor.
When performing the attack using the same method as for target 1, no fully correct key could be recovered with 200, 500, 1000, 2000, and 5000 traces per attacked round. Reducing the trigger window size of the attack on round one to only cover round one, like for the attack on AES for target 3, led to correct full-key recovery.
The attack with the smaller trigger window was run with 50, 100, 150, 200, 250, and 300 power traces. For each number of power traces, the CPA attack was repeated 50 times. The results of this can be found in Figure 15, 16 and Table 13.

Figure 15 shows the average success rate when running 50 CPA attacks with varying numbers of traces. At 200 traces, a success rate of 0.96 is achieved with a standard error of 0.028. The success rate then slightly decreases at 250 traces to 0.92 with a standard error of 0.038, to then return to 0.96 at 300 traces and a standard error of 0.028. Since there is an overlap in the error bars, the decrease at 250 traces is not meaningful, since the success rate cannot be distinguished within the shown uncertainty.

In Figure 16 can be seen that the capture and processing time both seem to follow a roughly linear increase. As with target 1 and target 2, the processing time is notably shorter than the capture time. As explained in the target 1 section of PRESENT 5.2.1, this most likely has to do with the

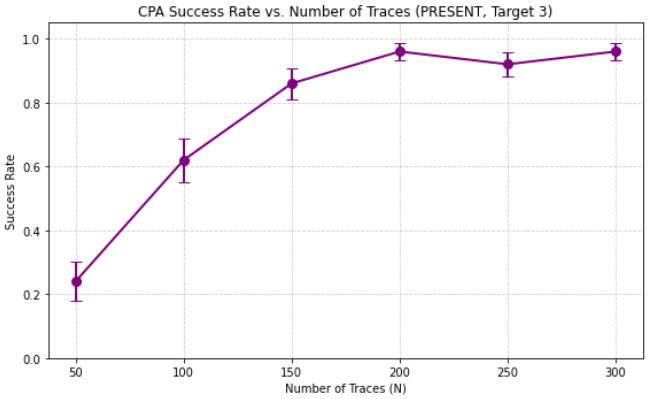

Figure 15: Success rate (over 50 iterations) for varying numbers of power traces per attacked round when attacking PRESENT on Target 3

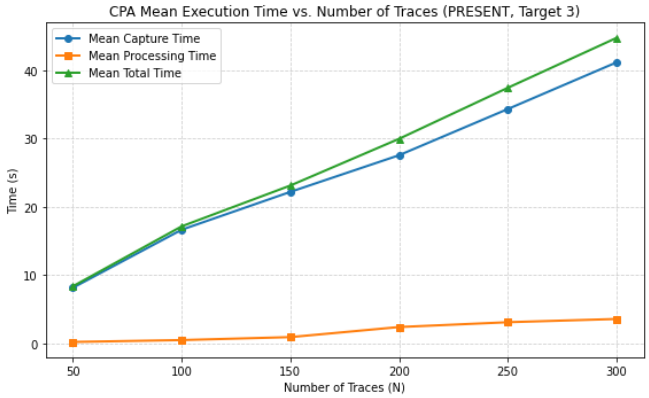

Figure 16: Average attack time of both attacked rounds (over 50 iterations) for varying numbers of power traces per attacked round when attacking PRESENT on Target 3

attack being performed by nibble instead of per byte.

From Figure 15 and Table 13, it can be seen that the lowest tested number of traces that reached a success rate of at least 0.95 was 200, with a success rate of 0.96 and a mean total capture and processing time of 29.95 seconds. Based on this, the success rate threshold of 0.95 was reached somewhere between 150 traces with a success rate of 0.86 and 200 traces with a success rate of 0.96, so the trace count threshold was bracketed by $(150, 200]$. However, due to the uncertainty in the estimated success rates, this interval should not be interpreted as an exact threshold for the underlying success probability. All reported trace counts for PRESENT are per attacked round.

Number of Traces (N)	**success Rate**	**Mean Round 1 Time (s)**			**Mean Round 2 Time (s)**			**Mean Total Time (s)**
		Capture	**Processing**	**Total**	**Capture**	**Processing**	**Total**	
50	0.24	4.04	0.11	4.15	4.12	0.11	4.23	8.38
100	0.62	8.30	0.25	8.55	8.35	0.25	8.60	17.15
150	0.86	11.03	0.46	11.49	11.16	0.47	11.63	23.12
200	0.96	13.96	1.20	15.16	13.59	1.20	14.79	29.95
250	0.92	17.30	1.54	18.84	17.02	1.57	18.59	37.43
300	0.96	20.71	1.79	22.50	20.43	1.80	22.23	44.73

Table 13: Results for different number of traces (over 50 iterations) on PRESENT on Target 3

5.2.4 Summary

The results of the experiments on PRESENT-80 show that CPA attacks are feasible on PRESENT on all three implementation targets. The reported number of traces for PRESENT refers to the captured number of traces per attacked round, meaning that the same number of traces was captured for the attack on round one and for the attack on round two.

Target 3 required the fewest traces to reach a success rate of at least 0.95, with 200 traces achieving a success rate of 0.96 and a mean total attack time of 29.95 seconds. However, target 3 used a

reduced trigger window for the attack on round one, while the other targets used a trigger window that covered the full encryption. Without this change in trigger window size, target 3 was not able to recover any fully correct keys with the tested number of traces. Therefore, the lower number of traces for target 3 cannot be attributed only to differences in implementation.
Target 1 required slightly more traces to reach the success rate threshold of at least 0.95, needing 300 traces to achieve a success rate of 0.98 and taking on average 19.23 seconds. Despite requiring more traces, the average total time for target 1 is shorter than the average total time for target 3. This difference can be explained by the average capture time for target 3 taking longer.
Target 2 required the most traces to reach a success rate of at least 0.95, needing 600 traces to achieve a success rate of 0.96 with a total mean attack time of 37.63 seconds.

Overall, under the tested configurations, target 3 required the fewest traces, followed by target 1, and finally target 2 required the most traces. However, the ranking of target 3 should be interpreted with the difference in trigger window in mind.

6 Discussion

This section discusses the results of the experiments and the differences observed between the ciphers and implementation targets. First, the success-rate results are discussed, followed by the number of traces required for successful key recovery and the measured runtime. Finally, the limitations of the experiments and comparison are discussed.

		AES	**PRESENT**
TARGET 1	**Success Rate**	0.96	0.98
	Number of Traces (N)	30	300
	Mean Capture Time (s)	0.92	8.22 (R1) + 8.56 (R2)
	Mean Processing Time (s)	1.10	1.23 (R1) + 1.22 (R2)
	Mean Total Time (s)	2.02	9.45 (R1) + 9.78 (R2)
TARGET 2	**Success Rate**	0.98	0.96
	Number of Traces (N)	400	600
	Mean Capture Time (s)	13.80	15.71 (R1) + 15.93 (R2)
	Mean Processing Time (s)	29.59	2.99 (R1) + 3.00 (R2)
	Mean Total Time (s)	43.39	18.70 (R1) + 18.93 (R2)
TARGET 3	**Success Rate**	1.00	0.96
	Number of Traces (N)	25	200
	Mean Capture Time (s)	1.08	13.96 (R1) + 13.59 (R2)
	Mean Processing Time (s)	0.91	1.20 (R1) + 1.20 (R2)
	Mean Total Time (s)	1.99	15.16 (R1) + 14.79 (R2)

Table 14: Number of traces, mean capture time, and mean processing time (over 50 iterations) for CPA with a success rate ≥ 0.95 for each experiment. R1, R2 indicate round one and round 2 statistics. For PRESENT, N is the number of traces captured per attacked round.

Success Rate For AES, targets 1 and 2 were able to achieve a success rate ≥ 0.95 using the same CPA attack method. Target 3 was unable to recover more than a few bytes of the key using the same attack method as used for targets 1 and 2. For a successful attack on target 3, the size of

the trigger window had to be reduced, only covering round 1 instead of the whole encryption. With that change, all targets were able to recover keys with a success rate ≥ 0.95.
For target 3, the full-encryption and reduced trigger-window configurations both used 5000 samples per trace and 4.00 ADC samples per target clock cycle, with no decimation. The full-encryption trigger window on target 3 lasted 85940 ADC clock cycles, while the reduced trigger window lasted 8668 ADC clock cycles. For the reduced trigger-window configuration, the start of the trigger was moved into the encryption function, closer to the start of round one. For the full-encryption trigger window, acquisition started outside of the encryption function right before it was called. Since only 5000 samples were stored in each trace, one possible explanation for why the reduced trigger window worked while the full-encryption trigger window did not is that operations before round one took up too much space in the stored trace, which was limited to 5000 samples.
From Table 6, it can be seen that round one on target 3 starts at approximately target cycle 1110. Since the 5000 stored ADC samples correspond to approximately 1250 target clock cycles, a trace starting at the beginning of the full-encryption trigger window would only store approximately the first 140 target cycles of round one. This corresponds to only about 5.3% of the full round. This might not have included enough of the SubBytes operations targeted by the CPA attack. In comparison, with the reduced trigger-window configuration approximately 47% of round one was stored. This could explain why the attack did not work with the full-encryption trigger window but did work after the trigger window was reduced.

For PRESENT, as with AES, targets 1 and 2 were able to recover correct full keys using CPA with a trigger window covering the whole encryption for the attack on round 1. Target 3 was not able to recover full correct keys with this trigger window, so the trigger window needed to be reduced to only round one of the encryption.
For target 3, the full-encryption and reduced trigger-window configurations both used 5000 samples per trace and 4.00 ADC samples per target clock cycle, with no decimation. The full-encryption trigger window on target 3 lasted 1529624 ADC clock cycles, while the reduced trigger window lasted 4144 ADC clock cycles. As with AES, reducing the trigger window changed which part of the execution was stored in the 5000-sample trace.
From Table 6, it can be seen that round one on target 3 starts at approximately target cycle 12331. Since the 5000 stored ADC samples correspond to approximately 1250 target clock cycles, a trace starting at the beginning of the full-encryption trigger window would end before round one was even reached. Therefore, the operations in round one would not have been stored in this trace. With the reduced trigger-window configuration, approximately 10.5% of round one was stored. Although this was only a small part of the full round, it was sufficient for successful key recovery. This also shows that the complete round did not need to be stored, as long as the stored interval contained enough of the operations targeted by the CPA attack.
For target 3, like for AES, reducing the trigger window on the same target changed the attack from not being able to recover keys to correct key recovery. With these changes, all targets were able to recover keys with a success rate ≥ 0.95. It should be noted that target 3 used a more targeted trigger-window configuration than targets 1 and 2. Therefore, the success-rate results for target 3 are not directly comparable to those of targets 1 and 2.

These results show that achieving a success rate ≥ 0.95 on different implementation targets was feasible, although target 3 required adjustments to the trigger window. The results also show that

the complete attacked round did not need to be stored for successful CPA, but the stored samples needed to include sufficient leakage from the targeted operations. The results suggest that both the cryptographic algorithm and the implementation target influence the effectiveness of the evaluated CPA attacks.

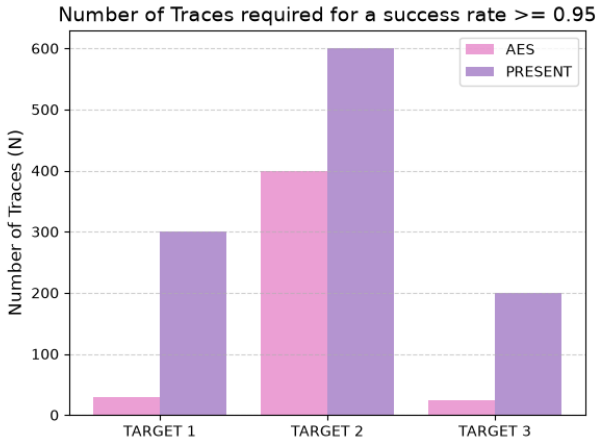

Figure 17: Number of Traces required for a success rate ≥ 0.95 for each target and cipher.

Number of Traces The number of traces used for comparison is the lowest tested value of N that reached a success rate of at least 0.95. For PRESENT, N is the number of traces used per attacked round, so $2N$ traces were captured in total because round one and round two were attacked separately. Since only specific values of N were tested, these values should not be interpreted as the exact minimum number of traces required.

From Table 14 and Figure 17 we determine that for target 1, the PRESENT attack needed 10 times as many traces as AES. For target 2, 1.5 times as many traces were needed for the PRESENT attack compared to the AES attack. For target 3, 8 times as many traces were needed for PRESENT compared to AES. For this target, reducing the trigger window made the attack go from not being able to recover a correct key to only needing 25 traces to achieve a success rate ≥ 0.95 for AES and 200 traces for PRESENT.
These comparisons use the lowest tested number of traces that reached the success-rate threshold and therefore should not be interpreted as exact ratios of the minimum number of traces required. For some neighboring values of N, especially for PRESENT on targets 2 and 3, the standard-error intervals overlap, which adds further uncertainty to the exact trace-count threshold. As discussed in the Success Rate paragraph, target 3 used a different trigger-window configuration for the AES and PRESENT attacks. This should be taken into account when comparing the results of target 3 with targets 1 and 2.

For all targets, the PRESENT attack required more traces than the AES attack to achieve a success rate ≥ 0.95. One possible explanation for this is the difference in the size of the attacked intermediate value. The AES attack used the HW of the output of an 8-bit S-box, while the

PRESENT attack used the HW of a 4-bit S-box output. The 4-bit S-box output has a HW between 0 and 4 and therefore predicts only half of the bits of the byte actually moved by the processor. The HW of the nibble value therefore predicts fewer bits than the AES leakage model. As a simplified approximation, predicting half as many bits can reduce the correlation by approximately $\sqrt{1/2} \approx 0.71$. A reduction in correlation by this amount could result in roughly twice as many traces being needed.
However, this explanation can only explain part of the observed difference. The ratios between the lowest tested trace counts that reached a success rate of at least 0.95 for AES and PRESENT were 10 times for target 1, 1.5 times for target 2, and 8 times for target 3. For targets 1 and 3, the increase is considerably larger than the roughly two times increase predicted by the approximation above.
The observation that attacks using 4-bit intermediate values required more traces than attacks using 8-bit intermediate values is consistent with the results of Biryukov et al. [BDG16]. In their experiments, AES, which uses an 8-bit S-box, required fewer traces than the evaluated ciphers using 4-bit S-boxes. However, their results also showed that implementation details and the selected intermediate value can influence the number of traces needed.
This is also consistent with Wang et al. [WYW+12], who found that their 8-bit model required fewer traces than their 4-bit model for attacking PRESENT. Their experiments were different from the ones in this thesis: they used a simulated hardware implementation, the Hamming distance leakage model, and only attacked PRESENT. However, their results still show that the size of the attacked intermediate value can influence the number of traces needed for CPA.
Since the simplified approximation cannot explain all of the additional traces required for a successful CPA attack on PRESENT, other factors could also contribute to the observed differences.

Figure 17 shows the difference in the number of traces needed between targets. For both AES and PRESENT, target 2 required more traces than both targets 1 and 3 to achieve a success rate $\geq$ 0.95. Both targets 1 and 2 use an ARM Cortex-M4 core, yet the number of traces needed for a successful attack differs considerably. Target 2 needed 13.3 times as many traces for AES and 2 times as many traces for PRESENT compared to target 1.
Even though targets 1 and 2 use the same processor core architecture and ISA, there are differences in the number of traces needed when attacking the same cipher using the same CPA attack and trigger window. One difference between the targets was the measurement setup. Target 1 was measured on a 3.3 V power rail through a 12 Ω shunt resistor, while targets 2 and 3 were measured on a 1.2 V power rail through a 10 Ω shunt resistor. These differences in the measurement setup could influence the measured power signal, which could cause differences in the number of traces needed for a successful CPA attack.
Other differences between the targets may also influence the measured side-channel leakage and therefore the effectiveness of the CPA attack. Examples of these differences are peripheral systems, power-management circuitry, differences in the power-supply path and board decoupling, Flash access or prefetch behavior, clock circuitry, and the overall target implementation. These factors were not investigated separately and therefore remain possible explanations rather than confirmed causes. As a result, the differences in the number of traces cannot be attributed to differences in the target implementations alone.
The influence of processor implementation on side-channel leakage is also supported by Barenghi et al. [BBIP21], who showed that microarchitectural components such as the register file, pipeline

registers, execution units, and load-store unit can contribute to side-channel leakage. They also found differences in leakage between processors with different pipeline and execution-unit structures, even though the processors used the same ISA. These findings show that processor microarchitecture is another possible factor that can influence side-channel leakage.

Target 3 differs further from targets 1 and 2 because it uses a RISC-V NEORV32 soft-core implemented on an FPGA instead of an ARM Cortex-M4 microcontroller. The measured side-channel leakage therefore also depends on the implementation of the soft-core.

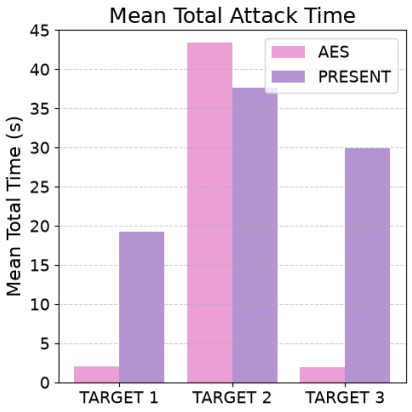

Figure 18: Mean total attack time for AES and PRESENT on all three targets at the lowest tested number of traces that achieved a success rate ≥ 0.95.

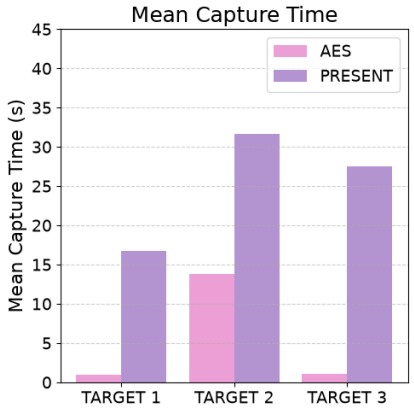

Figure 19: Mean capture time for AES and PRESENT on all three targets at the lowest tested number of traces that achieved a success rate ≥ 0.95. For PRESENT, the capture time is the total of the round-one and round-two attacks.

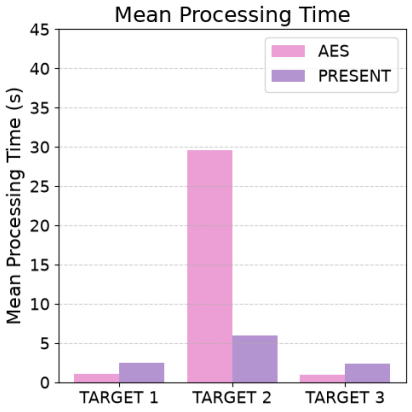

Figure 20: Mean processing time for AES and PRESENT on all three targets at the lowest tested number of traces that achieved a success rate ≥ 0.95. For PRESENT, the processing time is the total of the round-one and round-two attacks.

Runtime The mean total time can be seen in Figure 18. For targets 1 and 3, PRESENT had a longer total attack time than AES. For target 1, the mean total time was 2.02 s for AES and 19.23 s for PRESENT. For target 3, the mean total time was 1.99 s for AES and 29.95 s for PRESENT. For target 2, AES had a longer total attack time, with 43.39 s compared to 37.63 s for PRESENT. This difference is mainly caused by the larger processing time of AES on target 2.

The mean capture time can be seen in Figure 19. Attacks that needed more traces also had a higher capture time, since each additional trace required another encryption and trace capture. The capture time for PRESENT was longer than for AES on all targets. One reason for this is the larger number of traces required for PRESENT, which can be seen in Figure 17. In addition, the PRESENT attack attacked two rounds of the cipher using separate trace captures. This doubles the total number of traces that need to be captured, which further increases the capture time.

The mean processing time can be seen in Figure 20. For AES on targets 1 and 2, the processing time

was larger than the capture time, while for target 3 the processing time was slightly shorter than the capture time. The processing time for AES on target 2 was considerably larger than the capture time compared to the difference between those times for targets 1 and 3. From Tables 8, 9, and 10, it can be calculated that the processing cost was about 79 ms per trace for target 2, compared to 33 ms per trace on target 1 and 32 ms per trace on target 3. This difference was systematic across the tested numbers of traces. The exact cause of the larger processing time for target 2 was not determined.

For the PRESENT attacks, the processing time was considerably shorter than the capture time on all three targets. One possible explanation for this is the smaller number of hypothetical key values that need to be evaluated compared to AES because the PRESENT attack works per nibble instead of per byte. For AES, the key is recovered per byte (8 bits), while for PRESENT it is recovered per nibble (4 bits). For AES, 256 hypothetical key-byte values are tested for each of the 16 key bytes needed to recover the full key, resulting in 4096 values in total. For PRESENT, 16 hypothetical nibble values are tested for each of the 16 nibbles needed to recover a round key, and two round keys are needed, resulting in 512 values in total. This smaller number of values that need to be evaluated for PRESENT can help explain why the processing time for PRESENT is smaller than the capture time.

6.1 Limitations

One limitation is the fixed number of 5000 samples per trace. For target 3, the full-encryption trigger windows were considerably longer than the part of the execution that could be stored in these 5000 samples. As discussed in the Success Rate paragraph of the Discussion, this could have caused some of the first-round operations targeted by the CPA attack to fall outside the stored trace. Increasing the number of samples per trace could have allowed a larger part of the encryption to be recorded. Another possibility would have been to use decimation to cover a longer period of the execution. These alternatives were not tested.

For target 3, the trigger window needed to be reduced for AES and PRESENT to be able to recover keys. These changes in the attack configuration were needed to make the attacks work. As a result, target 3 used a more targeted trigger-window configuration, while targets 1 and 2 did not. This makes the results for target 3 less directly comparable to the results for targets 1 and 2.
Another limitation is that changing the trigger window in this way requires access to the target code in order to insert the trigger points and flash the modified implementation to the target. In a real-world attack where the attacker does not have access to the target code, it may not be possible to place the trigger directly around the targeted round. Instead, the attacker would have to estimate the location of the targeted operations from the observed power traces or other available information.

Another limitation is that the second-round attack on PRESENT recovers the full round key and not only the two bytes that are needed for reconstructing the full key. This results in additional processing and capture time that would not be necessary if only the missing bytes were recovered.

The three targets were not measured under identical electrical conditions. The main differences in the measurement setup can be seen in Table 15. Target 1 used a different supply voltage and

shunt resistor than targets 2 and 3. These differences could affect the measured power signal even if the targets were completely the same. Therefore, differences in attack performance cannot be attributed only to the implementation target. This is especially important for the considerably larger number of traces required for AES on target 2 compared to target 1.

	Supply voltage	**Shunt**	**ChipWhisperer version**
Target 1	3.3 V	12 Ω	5.6.1
Target 2	1.2 V	10 Ω	6.0.0
Target 3	1.2 V	10 Ω	6.0.0

Table 15: Measurement setup differences between the three implementation targets.

The ChipWhisperer software version also differed between the experiments. Target 1 was tested using ChipWhisperer 5.6.1, while targets 2 and 3 were tested using ChipWhisperer 6.0.0. Although the main acquisition parameters were kept the same, differences between the software versions or their default settings could have influenced the measurements.

Only one physical device was tested for each implementation target. Therefore, the results do not show whether the observed differences are representative of other devices of the same type.

AES and PRESENT were based on two different software code bases. Differences between these implementations could therefore also have influenced the comparison between the two ciphers.

The processing and timing measurements were performed in a virtual machine using two processor cores. Background system load, virtual-machine scheduling, and processor power-management settings could therefore have influenced the measured runtime.

6.2 Further Research

In this research, CPA on three different targets was evaluated. Future work could extend this comparison by evaluating more implementation targets. More microcontrollers could be compared, such as different ARM Cortex-M variants, for example, or different soft-cores on FPGAs. This could provide more information on what hardware characteristics have an effect on CPA.
Furthermore, the influence of scope settings could be investigated. The effect of settings such as trigger window size, sampling rate, or gain could be evaluated.
Other SCA metrics could be collected. In this research, the number of traces, success rate, and attack time were looked at. Other metrics to evaluate could be correlation, guessing entropy , and signal-to-noise ratio. These metrics could reveal more insight into the quality of the leakage.
Additionally, future research could investigate the impact of different software implementations. Protected implementations of the AES and PRESENT could be compared to unprotected ones. Optimized versions of these ciphers could also be compared against. The comparison could also be extended by including additional ciphers.
Finally, different side-channel analysis methods such as DPA and CNN analysis could be compared. Or also CPA with a different leakage model like Hamming distance compared to Hamming weight.

7 Conclusions

This research investigated the differences in CPA attacks on AES and PRESENT across three different implementation targets. The results show that both the cryptographic algorithm and the implementation target influence the effectiveness of the evaluated CPA attacks. After adapting the trigger window for target 3, all evaluated cipher and target combinations were able to achieve a success rate ≥ 0.95.

The number of traces needed differed between ciphers. PRESENT needed a larger number of traces than AES on all three targets to achieve a success rate ≥ 0.95. One possible explanation for part of this difference is that the AES attack used the HW of an 8-bit S-box output, while the PRESENT attack used the HW of a 4-bit S-box output. The PRESENT leakage model therefore predicts fewer bits than the AES leakage model. However, this alone could not explain all of the observed differences in the number of traces.

The results also show that the number of traces needed for a successful key recovery differs between targets. Target 2 required more power traces than target 1 and 3 for both AES and PRESENT to achieve a success rate ≥ 0.95. This shows that the implementation target has an influence on the number of traces needed.
Furthermore, targets 1 and 2 use an ARM Cortex-M4 core, yet target 2 required significantly more traces for a successful attack. This suggests that implementation-specific characteristics of the hardware, peripheral systems, power-management circuitry, and overall chip implementation have an influence on the power leakage and thus influence the effectiveness of CPA attacks. However, differences in the measurement setup could also have contributed to the observed differences between the targets.

For target 3, the trigger window had to be reduced for both AES and PRESENT so full keys could be recovered. The measurements showed that successful key recovery was still possible even when only part of the attacked round was stored in the power trace. This suggests that storing the complete round was not necessary, as long as the stored samples contained sufficient leakage from the operations targeted by CPA. This shows that differences between implementation targets can require changes to the attack configuration and not only a different number of traces. The need to reduce the trigger window also shows that differences between implementation targets can also require changes to the attack configuration and not only a different number of traces. However, target 3 therefore used a more targeted trigger-window configuration than targets 1 and 2, which limits how directly its results can be compared with the other targets.

Overall, the results show that CPA attack performance differed both between the evaluated ciphers and between implementation targets. These differences were observed in the number of traces required, the runtime of the attacks, and the attack configuration needed for successful key recovery. However, differences in the measurement setup and attack configuration should be taken into account when interpreting the cross-target comparison.

References

[Atm15] Atmel Corporation. *SAM4S Series: 32-bit Cortex-M4 Microcontroller Datasheet*. Atmel Corporation, rev. atmel-11100k edition, June 2015. https://www.st.com/resource/en/datasheet/stm32f303rc.pdf.

[BBIP21] Alessandro Barenghi, Luca Breveglieri, Niccolo Izzo, and Gerardo Pelosi. Exploring cortex-m microarchitectural side channel information leakage. *IEEE Access*, 9:156507–156527, 2021.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *International workshop on cryptographic hardware and embedded systems*, pages 16–29. Springer, 2004.

[BDG16] Alex Biryukov, Daniel Dinu, and Johann Großschädl. Correlation power analysis of lightweight block ciphers: From theory to practice. In *International Conference on Applied Cryptography and Network Security*, pages 537–557. Springer, 2016.

[BKL+07] Andrey Bogdanov, Lars R Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew JB Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. Present: An ultra-lightweight block cipher. In *International workshop on cryptographic hardware and embedded systems*, pages 450–466. Springer, 2007.

[DBN+01] Morris J Dworkin, Elaine Barker, James R Nechvatal, James Foti, Lawrence E Bassham, E Roback, James F Dray Jr, et al. Advanced encryption standard (aes). 2001.

[DL21] Maurizio Di Lorenzo. *Comparison between Differential and Correlation Power Analysis Attacks on Embedded Systems*. PhD thesis, Politecnico di Torino, 2021.

[DR99] Joan Daemen and Vincent Rijmen. Aes proposal: Rijndael. 1999.

[Fre] Free Software Foundation. *Using the GNU Compiler Collection (GCC): Options That Control Optimization*. GCC 8.3.0.

[JTS+20] Om Jena, A. Tripathy, S. Swagatam, Smita Rath, and Alok Tripathy. Dual encryption model for preserving privacy in cloud computing. *Advances in Mathematics: Scientific Journal*, 9:6667–6678, 08 2020.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Annual international cryptology conference*, pages 388–397. Springer, 1999.

[Kok19] Kokke. tiny-aes-c: Small portable aes128/192/256 in c. https://github.com/kokke/tiny-AES-c, 2019.

[Lat25] Lattice Semiconductor. iCE40 UltraPlus Family Data Sheet. https://www.latticesemi.com/view_document?document_id=51968, 2025.

[LBC17] Owen Lo, William J Buchanan, and Douglas Carson. Power analysis attacks on the aes-128 s-box using differential power analysis (dpa) and correlation power analysis (cpa). *Journal of cyber security technology*, 1(2):88–107, 2017.

[LBC18] Owen Lo, William J Buchanan, and Douglas Carson. Correlation power analysis on the present block cipher on an embedded device. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–6, 2018.

[New] NewAE Technology Inc. tiny-aes128-c. https://github.com/newaetech/chipwhisperer/tree/5.6.1/firmware/mcu/crypto/tiny-AES128-C. Source code included with ChipWhisperer 5.6.1.

[New25a] NewAE Technology. Cw1173 chipwhisperer-lite. https://chipwhisperer.readthedocs.io/en/latest/Capture/ChipWhisperer-Lite.html, 2025. ChipWhisperer Documentation. Accessed: 31 July 2026.

[New25b] NewAE Technology. Cw303 arm target. https://chipwhisperer.readthedocs.io/en/latest/Targets/CW303%20Arm.html, 2025. ChipWhisperer Documentation. Accessed: 31 July 2026.

[New25c] NewAE Technology. Cw312t-ice40 target board documentation. https://chipwhisperer.readthedocs.io/en/latest/chipwhisperer-target-cw308t/CW312T_ICE40UP/README.html, 2025. Accessed: 31 July 2026.

[New25d] NewAE Technology. Cw312t-sam4s target board documentation. https://chipwhisperer.readthedocs.io/en/latest/chipwhisperer-target-cw308t/CW312T_ATSAM4S/README.html, 2025. Accessed: 31 July 2026.

[New25e] NewAE Technology. Cw313 – 20-pin to card edge (cw312 style) breakout/adapter board. https://chipwhisperer.readthedocs.io/en/latest/Targets/CW313.html, 2025. ChipWhisperer Documentation. Accessed: 31 July 2026.

[New25f] NewAE Technology Inc. Chipwhisperer neorv32 ice40 pre-build core. https://github.com/newaetech/chipwhisperer/tree/develop/software/chipwhisperer/hardware/firmware/cwtargetice40, 2025. Accessed: 2026-08-07.

[New25g] NewAE Technology Inc. Chipwhisperer: The complete open-source toolchain for side-channel power analysis and glitching attacks. https://github.com/newaetech/chipwhisperer, 2025.

[New26] NewAE Technology Inc. Simpleserial documentation. https://chipwhisperer.readthedocs.io/en/latest/simpleserial.html, 2026.

[Nol20] S. Nolting. The neorv32 risc-v processor. https://github.com/stnolting/neorv32, 2020.

[PGA+23] Kostas Papagiannopoulos, Ognjen Glamočanin, Melissa Azouaoui, Dorian Ros, Francesco Regazzoni, and Mirjana Stojilović. The side-channel metrics cheat sheet. *ACM Computing Surveys*, 55(10):1–38, 2023.

[Sha49] Claude E Shannon. Communication theory of secrecy systems. *The Bell system technical journal*, 28(4):656–715, 1949.

[STM18] STMicroelectronics. *STM32F303xB STM32F303xC: Arm Cortex-M4 32-bit MCU with FPU, up to 256 KB Flash, 48 KB SRAM, Datasheet*. STMicroelectronics, rev. 14 edition, October 2018. https://www.st.com/resource/en/datasheet/stm32f303rc.pdf.

[SZSM24] Abolfazl Sajadi, Nusa Zidaric, Todor Stefanov, and Nele Mentens. A systematic comparison of side-channel countermeasures for risc-v-based socs. In *2024 IEEE Nordic Circuits and Systems Conference (NorCAS)*, pages 1–7. IEEE, 2024.

[Ton19] (Pepton21) Petar Tonkovic. present-cipher. https://github.com/Pepton21/present-cipher, 2019.

[WYW+12] Chenxu Wang, Mingyan Yu, Jinxiang Wang, Peihe Jiang, and Xiaochen Tang. A more practical cpa attack against present hardware implementation. In *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, volume 3, pages 1248–1253. IEEE, 2012.