Journal of Cyber Security Technology

ISSN: 2374-2917 (Print) 2374-2925 (Online) Journal homepage: www.tandfonline.com/journals/tsec20

Power analysis attacks on the AES-128 S-box using differential power analysis (DPA) and correlation power analysis (CPA)

Owen Lo, William J. Buchanan & Douglas Carson

To cite this article: Owen Lo, William J. Buchanan & Douglas Carson (2017) 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, DOI: 10.1080/23742917.2016.1231523

To link to this article: https://doi.org/10.1080/23742917.2016.1231523

JOURNAL OF CYBER SECURITY TECHNOLOGY, 2017
VOL. 1, NO. 2, 88–107
https://doi.org/10.1080/23742917.2016.1231523

Taylor & Francis
Taylor & Francis Group

Power analysis attacks on the AES-128 S-box using differential power analysis (DPA) and correlation power analysis (CPA)

Owen Lo [a], William J. Buchanan[a] and Douglas Carson[b]

[a]The Cyber Academy, Edinburgh Napier University, Edinburgh, UK; [b]Keysight Technologies, Edinburgh, UK

ABSTRACT

This article demonstrates two fundamental techniques of power analysis, differential power analysis (DPA) and correlation power analysis (CPA), against a modern piece of hardware which is widely available to the public: the Arduino Uno microcontroller. The DPA attack we implement is referred to as the Difference of Means attack while the CPA attack is implemented by building a power model of the device using the Hamming Weight Power Model method. The cryptographic algorithm we have chosen to attack is AES-128. In particular, the AddRoundKey and SubBytes functions of this algorithm are implemented on an Arduino Uno and we demonstrate how the full 16-byte cipher key can be deduced using the two techniques by monitoring the power consumption of the device during cryptographic operations. The results of experimentation find that both forms of attack, DPA and CPA, are viable against the Arduino Uno. However, it was found that CPA produces results which are easier to interpret from an analytical perspective. Thus, our contributions in this article is providing a side-by-side comparison on how applicable these two power analysis attack techniques are along with providing a methodology to enable readers to replicate and learn how one may perform such attacks on their own hardware.

ARTICLE HISTORY
Received 29 August 2016
Accepted 30 August 2016

KEYWORDS
Power analysis; Side Channel Attacks; AES; DPA; CPA

1. Introduction

With the technological growth and increased interest on the Internet of Things (IoT), concerns over hardware security are now on the rise. With this growing concern, recent research is now focused on exploring both the attack and defensive aspect of such devices including microcontrollers, sensors, wearable gadgets among other similar hardware. Naturally, to achieve the goal of building defensive capabilities in hardware, one must first understand and be aware of the exploits it is susceptible to. One such exploit which may be applied to hardware security is Side Channel Attacks (SCA). SCA involves monitoring the external outputs of the hardware while cryptographic operations are running with the goal of attempting to information which would result in the security of

CONTACT Owen Lo o.lo@napier.ac.uk The Cyber Academy, Edinburgh Napier University, Edinburgh, UK

the device being compromised. Common external outputs which may be monitored include heat, sound, time, electromagnetic radiations and power consumption.

In the scope of this article, we focus on attacks based on monitoring the power consumption of a device. In particular, the aim of this article is to present the findings on performing power analysis attacks against the AES-128 algorithm by monitoring the power consumption of a microcontroller (Arduino Uno [1]) while cryptographic operations take place. We analyse and compare two commonly discussed power analysis techniques, namely 'Difference of Means' and 'Hamming Weight Power Model' attack. The former technique falls under the category of a differential power analysis (DPA) attack while the latter is categorised as a correlation power analysis (CPA) attack.

The contribution this article makes is to compare and contrast how applicable these two power analysis attack techniques are along with providing a methodology to enable readers to replicate and learn how one may perform such attacks on their own hardware. This article may be of particular interest to researchers who are just starting work in this area of research since snippets of code are provided to enable one to apply the theories of power analysis in practice.

2. Background

2.1. *Overview of Side Channel Attacks*

A SCA is an attack on the information of a cryptographic device. The concept of 'information' in the context SCA generally refers to a secret key of a cryptographic algorithm. A SCA is carried out by monitoring the physical outputs of a device (e.g. power consumption, time taken to carry out an operation, emission of heat, light and sound). The hypothesis made in such an attack is that the physical outputs of a cryptographic device demonstrate correlation with the internal state of the device when conducting cryptographic operations. Figure 1 depicts an abstraction of this concept.

Rather than attempt to break the core implementation of a cryptographic device, SCA aims to monitor 'leaked' information produced by a device during its normal operation and attempt to deduce the secret key leakage gathered. The work of Zhou and Feng [2] lists over 10 different types of SCA which has been presented in research. Some of the most common attacks include timing attacks [3], fault attacks [4] and power analysis

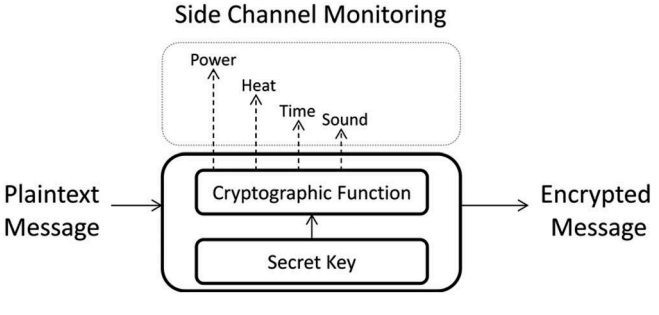

Figure 1. Side channel monitoring.

attacks. For the scope of this article, we focus on power analysis attack, which is focused on monitoring the power consumption of a device in order to deduce information leakage.

2.2 *Power analysis techniques*

Power analysis attacks are carried out by monitoring the power consumption on a cryptographic device. A common piece hardware which can be used to monitor the device is an oscilloscope. In this type of attack, one must first assume that there is correlation between the level of power consumption and cryptographic operations of the device. Originally, there were two main categories of power analysis attacks including simple power analysis (SPA) and DPA. As research progressed in the area of SCA, a third category was found in literature which is formally referred to CPA.

In SPA, one monitors the power trace of a cryptographic device (as it performs a cryptographic function) and attempts to determine the secret key based on the measurement (e.g. voltage levels) produced. In practice, it can be rather difficult to deduce the values of a secret by SPA alone. However, as Kocher et al. [5] have shown, SPA can still be potentially useful for gathering information about the type of algorithm. To provide an example of SPA, Figure 2 presents a power trace captured from an Arduino Uno as it runs a single AES-128 cryptographic operation. In AES-128, there are 10 rounds of operation which can be identified as the 10 nadirs in the figure presented. Thus, although one is unable to deduce the secret key using this technique, it does present the capability to identify the cryptographic algorithm running on device and enable more powerful attacks which specifically exploit any weakness of an algorithm to take place.

An example of a more power attack is DPA. This attack makes use of statistical techniques to identify *differences* in power traces, thus revealing data leakage which may result in the correct secret key being guessed. CPA, on the other hand, aims to reveal data leakage by finding relationships between characteristics of power traces and a hypothesised power model. If correlation is found between these two variables, one has the capability to predict the correct secret key if enough power traces are gathered.

Numerous variations of DPA and CPA attacks have been proposed in literature. To achieve the aim of article, a comparison between DPA and CPA attacks in practice, we focus on experiments to compare one of the first original DPA attacks which is referred

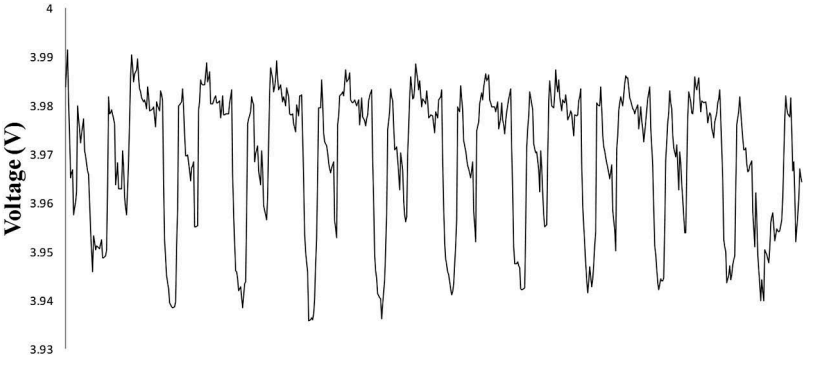

Figure 2. Simple power analysis on AES-128.

to as Difference of Means as described in [6] against the simplest, in our opinion, CPA attack referred to as Hamming Weight Power Model as first designed by [7].

2.2.1. *DPA: Difference of Means*

In DPA, the hypothesis is that small variations in power level may be observed in a trace based on the output of an encryption algorithm. An example of this is to observe the least significant bit (LSB) of an output. The LSB is the unit bit of a binary number. For example, the LSB of binary value 11001101 is 1. If the LSB output produced by an encryption algorithm is 1 then, in theory, the device under test should consume more power in comparison with a 0 value. If the hypothesis holds true, we may exploit this fact to deduce the cipher key during cryptographic operations.

It must be emphasised that this variation in power consumption is very small and almost impossible to detect by the naked eye. Thus, in DPA, one must gather a large number of traces, sort them into two subsets and calculate the average of each subset. This technique is referred to as the Difference of Means attack. The difference in means between each subset allows one to deduce whether there is any significance in the proposed hypothesis. In the case of significance, the difference in averages between the two subsets will highlight the variation in power consumption when a LSB of 0 is compared against an LSB of 1 (thus proving the hypothesis). However, if the subsets are not sorted correctly, no significance should be found.

Sorting of traces is formally referred to as the selection function but, ultimately, it comes down to an educated guess when deciding which subset a trace belongs to. Referring back to the concept of LSB, the selection function criterion for this example would be to sort traces into a subset where it is believed that all traces produce the LSB of 0 and a second subset where all traces produce the LSB of 1.

Once gathering and sorting of traces into two separate subsets is complete, the average of each subset is calculated in a point-by-point basis. The difference in averages can then be calculated by simply subtracting the points of the first subset against the points of the second subset. If the subsets are sorted correctly, a line plot of this difference of averages will result in unique peaks and nadirs easily discernible to the naked eye. On the other hand, if the subsets were sorted incorrectly, the process averaging and subtracting the difference in averages will cancel out each subset and results in a line plot close to 0 (although in practice, the values will never be exactly 0 due to noise and other interferences).

To mount such an attack in practice, one must have some form known variable whether it is an input or an output. An example of an input may be known plaintext values while an example of a output could be the ciphertext values produced by the cryptographic algorithm. By having a known input or output, the selection function can be implemented to sort power traces into the two subcategories based on hypothesis of how the known input will affect variations in power level (e.g. will LSB 1 or will LSB 0 be produced?) during key stages of a cryptographic operation which involve or relate back to the cipher key.

As originally presented and described in [6], the Difference of Mean attack can be expressed as Equation 1. In this equation, T is each individual power consumption trace while T_i is the ith trace. The variable j represents the power consumption value at time jth offset. C represents the known inputs or outputs during an attack with C_i being

relative to the *i*th trace. The selection function itself is represented as $D(C_i, K_n)$ which takes in the parameters of our known input or output C_i along with the key guess K_n where n is relative to one of the cipher keys we are attempting to deduce (n would be 16 in the case of AES-128). The top fraction of this equation represents the first subcategory of traces (as sorted by the selection function) while the bottom represents the second category of traces. The significant difference Δ_D for each point j is calculated by subtracting the average of the first subcategory against the second subcategory:

$$\Delta_D[j] = \frac{\sum_{i=1}^{m} D(C_i,K_n)T_i[j]}{\sum_{i=1}^{m} D(C_i,K_n)} - \frac{\sum_{i=1}^{m}(1-D(C_i,K_n)T_i[j])}{\sum_{i=1}^{m}(1-D(C_i,K_n))} \quad (1)$$

2.2.2. *CPA: Hamming Weight Power Model*

In CPA, the goal is to accurately produce a power model of the device under attack. During an attack, the aim is to find correlation between a predicted output and the actual power output of a device. If the power model is accurate then a strong correlation should be demonstrated between the predicted output and actual output. If this correlation is found then, similar to DPA, gathering a large number of traces will enable one to show that the correctly predicted cipher key will demonstrate the highest level of correlation.

One power model which may be used is the Hamming Weight Power Model. Traditionally, the Hamming weight of a value is the number of non-zeroes. For example, in the binary number 1100 0010 the Hamming weight would be 3. The assumption in using the Hamming Weight Power Model in power analysis attacks is that the number of bits set to 0 or 1 of an output is correlated with the power consumption of a device. The Hamming weight itself is then used as an arbitrary unit to model the consumption of power in a device. Hamming weight units can then be compared to the actual voltage levels of power traces captured when a device was performing cryptographic operations. This act of comparison is the process of finding correlation between the modelled power unit values and the actual power consumed.

One technique to calculate correlation between the power model and the actual power consumption is to use Pearson correlation coefficient equation. In essence, this equation will take two sets data (W and P) and calculate whether there is a linear (positive or negative) correlation between the two sets of values. We may use this equation to find significance in our power traces since the assumption with the Hamming Weight Power Model is that as the number of 1's increase in our predicted output, so too does the power consumption increase in the actual output (and vice versa).

Table 1 gives an example of possible outputs produced by a cryptographic device and the power consumption measured during each output (note that these are not real-life values but merely serve as an example). The Hamming weight for output 0000 0111 would be 3 while 0000 1111 would be 4 and so on. Assume we produce a Hamming Weight Power Model which is completely accurate then, as demonstrated in Table 2, a strong positive correlation will be found (i.e. as our power consumption increases so too does the number of 1's in our binary output).

Table 1. Example of power consumption against output.

Power consumption	Output
4.15	0000 0111
4.15	0000 0111
4.20	0000 1111
4.21	0000 1111
4.24	0001 1111
4.25	0001 1111
4.26	0001 1111
4.30	0011 1111
4.31	0011 1111
4.29	0011 1111

Table 2. Example of correlation coefficient computation.

Power consumption (W)	Predicted hamming weight unit of output (P)	Correlation coefficient (ρ) of W and P
4.15	3	0.9919
4.15	3	
4.20	4	
4.21	4	
4.24	5	
4.25	5	
4.26	5	
4.30	6	
4.31	6	
4.29	6	

Similar to the Difference of Means attack described in Section 2.2.1, there is also a need to have some form of known input or output when performing an attacking using this CPA technique. Having a known variable is required since the Hamming Weight Power Model itself is built by hypothesising how many bits are set to 1 as a form of output during key stages of a cryptographic operation which involve or relate back to the cipher key. Originally presented in [8], the correlation coefficient (ρ) between actual power consumption and predicted Hamming weight power can be calculated using Equation 2. The variable W represents a set of real power consumption values while P is a set of predicted Hamming weight values. The Cov is the covariance operation while Var is the variance operation:

$$\rho(W,P) = \frac{Cov(W,P)}{\sqrt{Var(W)}\sqrt{Var(P)}} \tag{2}$$

2.3. *AES-128 algorithm overview*

The final part of the background section of this article details the inner workings of the AES-128 algorithm's AddRoundKy and SubBytes function. Numerous resources including [9] already provided a full description of the AES algorithm; therefore, focus is only given on the aspects of AES which are relevant for the attacks demonstrated in this article. Although the attacks presented are catered towards AES-128, the theory may also apply to other implementations too (AES-192 and AES-256).

2.3.1. *AddRoundKey and SubBytes steps*

During the initialisation stage of AES, only one step is performed: AddRoundKey. Next, the algorithm will move on to the first round of operation and perform the first step: SubBytes. A description of these two steps is provided.

At the start of encryption, the plaintext values (the data to be encrypted) and the cipher key values (the key used for encryption and decryption purposes) will be each arranged into a 4×4 matrix in the positions as shown in Figure 3. Each value in this matrix holds 1 byte of data. During the AddRoundKey step, each plaintext value is exclusive OR'd with a cipher key value at a matching position in the 4×4 matrix. Thus, the exclusive OR operations which take place can be represented in Equation 3. Or, more simply, it may be represented as shown in Equation 4 where $_i$ is a position in the 4×4 matrix.

$$[P_0 \oplus K_0, P_1 \oplus K_1, P_2 \oplus K_2 \ \cdots \ P_{15} \oplus K_{15}] \tag{3}$$

$$P_i \oplus K_i \tag{4}$$

After AddRoundKey, the SubBytes step will use the result produced by $P_i \oplus K_i$ as a lookup for a value stored in the Rijndael S-box. The S-box output will replace $P_i \oplus K_i$. In other words, the output for each $P_i \oplus K_i$ S-box lookup result can be represented as Equation 5 where O_i is each individual output and S is the S-box lookup operation.

$$O_i = S[P_i \oplus K_i] \tag{5}$$

The S-box is a 16×16 matrix of values which remains constant for all AES implementations. Each position in the 16×16 matrix will hold 1 byte of data. The overall purpose of the S-box is to enable greater 'mixing' of data values to take place therefore ensuring the encryption algorithm is more computationally difficult to break. Figure 4 presents the Rijndael S-Box. The lookup of a value is performed by taking the first 4 bits of $P_i \oplus K_i$ as the column position and the last 4 bits of $P_i \oplus K_i$ as the row position. As an example, if $P_i \oplus K_i$ produced the hex value A1, the S-box lookup operation would replace this with hex value 32.

To provide concise understanding of the operations of AddRoundKey and SubBytes, a fully worked out example is provided. Firstly, assume our plaintext values (P), represented in hex format, are as follows: 01, 01, 02, 02, 03, 03, 04, 04, 05, 05, 06, 06, 07, 07, 08, 08. Next, the cipher key values (K) are as follows: A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, AA, AB, AC, AD, AE, AF. Using these example values, the two 4×4 matrixes constructed and result produced by the AddRoundKey is shown in Figure 5, while Figure 6 shows the S-box output produced by the SubBytes step.

In the context of this article, the DPA and CPA attacks implemented aim to exploit the fact that information may be leaked if one was to monitor the power consumption of a

P_0	P_4	P_8	P_{12}
P_1	P_5	P_9	P_{13}
P_2	P_6	P_{10}	P_{14}
P_3	P_7	P_{11}	P_{15}

$\oplus$

K_0	K_4	K_8	K_{12}
K_1	K_5	K_9	K_{13}
K_2	K_6	K_{10}	K_{14}
K_3	K_7	K_{11}	K_{15}

Figure 3. Plaintext and cipher key arrangement.

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1x	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2x	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3x	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4x	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5x	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6x	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7x	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8x	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9x	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
ax	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
bx	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
cx	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
dx	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
ex	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
fx	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 4. Rijndael S-box.

01	03	05	07	⊕	A0	A4	A8	AC	=	A1	A7	AD	AB
01	03	05	07		A1	A5	A9	AD		A0	A6	AC	AA
02	04	06	08		A2	A6	AA	AE		A0	A2	AC	A6
02	04	06	08		A3	A7	AB	AF		A1	A3	AD	A7

Figure 5. Example of AddRoundKey $P_i \oplus K_i$.

A1	A7	AD	AB	SubBytes →	32	5C	95	62
A0	A6	AC	AA		E0	24	91	AC
A0	A2	AC	A6		E0	3A	91	24
A1	A3	AD	A7		32	0A	95	5C

Figure 6. Example of SubBytes $S[P_i \oplus K_i]$.

cryptographic device during the point in which the S-box lookup is carried out. Section 3 details the design and implementation steps to enable such power analysis attacks to take place.

3. Design and implementation

Fundamentally, to mount an attack using either of the two power analysis techniques described in Section 2, there is the requirement for an attacker to have some form of adjustable variable which is both known and controllable (e.g. plaintext inputs or ciphertext outputs). By having a known variable, predictions can be for LSB output (for Difference of Means attack) or predictions can be made for power consumption (via the Hamming Weight Power Model attack). In the attacks implemented, the control we have chosen is plaintext values while the unknown variable is the cipher key of the AES-128 implementation.

3.1. *Hardware and software*

The following hardware and equipment was used for the implemented attacks:

- **Arduino Uno**: The device under test which runs the AddRoundKey and SubBytes operations of AES-128. This microcontroller uses the ATMega328P chip.
- **Keysight MSOX4104A**: The oscilloscope which was used to gather power traces. The oscilloscope includes four analog channels with a maximum of 1 GHz bandwidth and 5 GSa/s sample rate [10].
- **Keysight N2894A Probe**: The passive probe used to monitor power traces.
- **Keysight 1160A Probe**: The passive probe used to monitor Port 13 on the Arduino Uno. This probe is used as a trigger to let the oscilloscope know when to capture power traces (see Section 3.2 for further details).
- **Apple Macbook Pro Laptop**: Used as the client to enable control of the Arduino Uno along with interfacing with the oscilloscope to gather and save traces. The laptop is equipped with a 3.1 GHz Intel Core i7, 16 GB RAM and a 1 TB SSD.

The following software is used for this attack:

- **Arduino Software IDE**: Arduino's bespoke programming language is used to implement and program the microcontroller with the AddRoundKey and SubByte steps of AES.
- **Microsoft .NET C#**: Used to write client side software which runs on the Macbook Pro. Code was written to interface with, control and retrieve power traces from the oscilloscope using Standard Commands for Programmable Instruments (SCPI). Additionally, this language was used to also implement the Difference of Means and Hamming Weight Power Model attacks to be mounted against the power traces gathered.

3.2. *Data capture workflow*

Figure 7 presents a high-level workflow of the data capture process. A description of each step in the workflow follows:

(1) A command is sent from the laptop to the Arduino Uno through serial communication to signal it to run the AddRoundKey and SubBytes step against a known plaintext value. The AddRoundKey and SubBytes steps are ran 10 times for each individual plaintext value to enable the oscilloscope to get an average result (and thus help mitigate background noise and interference).
(2) The oscilloscope is set to capture each trace using the 'Average' function [11] so that an average of 10 traces is computed for each unique step of AddRoundKey and SubBytes. Triggering and capturing of the power trace is achieved by monitoring port 13 on the Arduino which is set to high when the cryptographic functions are running. The actual power consumption is monitored by inserting

a 100 Ω in series with VCC pin (pin 7) of the ATmega328p chip and probing the pin as shown in Figure 8.

(3) Once 10 traces have been captured by the oscilloscope, the laptop will retrieve and save the averaged result in as a CSV file on its disk drive. Step 1 to Step 3 are reiterated for each plaintext value from 00 to FF.

(4) After having gathered 256 averaged data traces of plaintext values applied to the AddRoundKey and SubBytes step of the Arduino, offline analysis can then take place to compute the results of key guesses using both the Difference of Means and Hamming Weight Power Model technique.

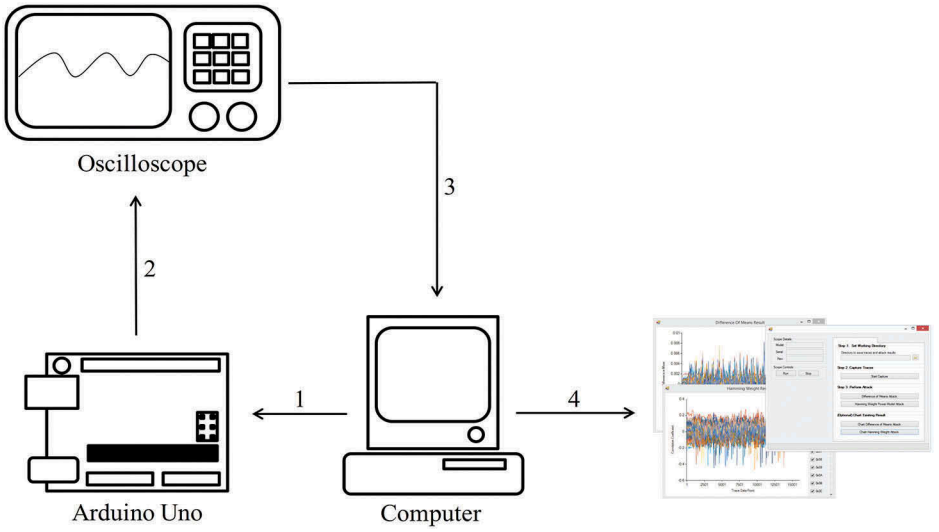

Figure 7. Data capture workflow.

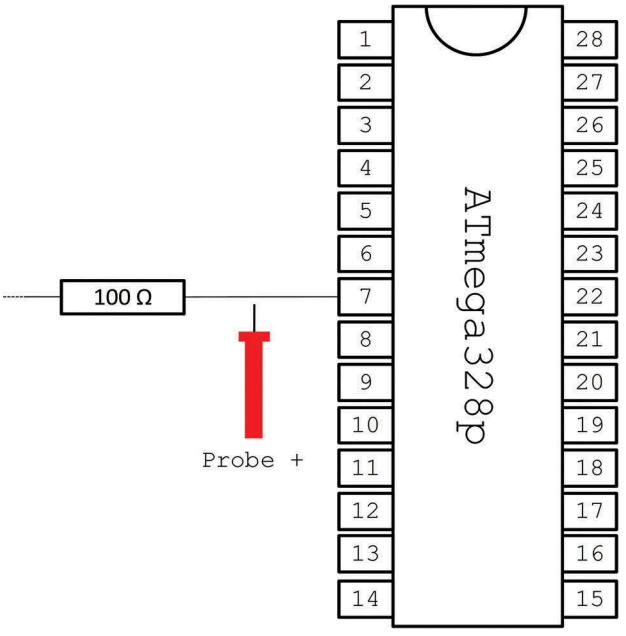

Figure 8. Probe point on Arduino Uno ATmega328p.

3.3. *AES-128 Arduino Uno implementation*

To demonstrate and compare capabilities of the Difference of Means and Hamming Weight Power Model attack, the Arduino Uno implements the AddRoundKey step which occurs during the initialisation stage of AES-128 along with the round one SubBytes step. Pseudocode to illustrate how these two steps may be implemented on the Arduino Uno is presented below. Text in **bold** represent variables while index assignment or lookup of an array is represent by the opening and closing square brackets [].

```
set plainText to a known value

for i 0 to 9:
  delay 500
  set LED 13 to HIGH
       for j 0 to 15:
            sbox_lookup[j] = s[plainText[j] XOR key[j]]
  set LED 13 to LOW
```

The variable s is an array of 256 bytes and consists of the S-Box entries 0×63 to 0×16 while the `sbox_lookup` is an array of 16 bytes which stores the S-box lookup result. The `plainText` variable is an array of 16 bytes which contains our plaintext input. Each plaintext value is known during an attack.

As demonstrated in the pseudocode, each plaintext value is operated 10 times in order for an average to be acquired. Furthermore, plaintext values should be known or observable by the attacker. In our implementation, we chose to increment `plaintText[]` by one each time this routine occurs (e.g. plain text array will start at 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, and end at FF, FF, FF, FF, FF, FF, FF, FF, FF, FF, FF, FF, FF, FF, FF, FF) A delay of 500 ms is implemented before the next iteration occurs to ensure the oscilloscope has enough time to capture the trace. The `sbox_lookup[]` array is used to store the result of the AddRoundKey and SubBytes operation while s is a 256 byte array which contains the constants of Rijndael S-Box.

Naturally, given that this implementation of AddRoundKey and SubBytes is bespoke, one may question the accuracy of the results produced. Thus, validation of the SubBytes output of our implementation was compared against results provided by FIPS-197 [12] using their test vectors. Both the output produced by our implementation and FIPS-197 were the same.

3.4. *Implementing Difference Of Means (DPA) attack*

To implement the Difference of Means attack, we apply the hypothesis that there should be a significant difference in power consumption if the LSB of an output is 0 against an output of 1. The output we wish to predict is the S-box lookup produced by the Arduino Uno relative to known plaintext values.

In order to predict the LSB of outputs, a model of the S-box is produced on the software running the attack which replicates the AddRoundKey and SubBytes function of the Arduino Uno. This attack software was written in .NET C# due to familiarity with

this high-level programming but any other language may also be applicable. Since both the order and actual plaintext values which were used by the Arduino Uno during the AddRoundKey and SubBytes function are known, we can sort each S-Box output against a key guess into two subsets (subset 1 where LSB is 0 and subset 2 where LSB is 1). By testing every possible key value, 0–255, against our known plaintext values, also 0–255, the correct key should produce the highest significant difference. The pseudocode which follows provides an example how one may implement this attack using a programming language of their own choosing. Along with the pseudocode conventions discussed previously, code which is presented in *italics* represent methods or functions which require implementation.

```
for KeyGuess 0 to 255:
  for plaintText 0 to 255:
        sBox_lookup = s[plainText XOR keyGuess]

        if (LastSignificantBit(sBox_lookup) == 1):
              Add trace relative to plaintText input to subset1
        else if (LastSignificantBit(sBox_lookup) == 0):
              Add trace relative to plainText input to subset0

getAverage(subset0)
getAverage(subset1)
calculatePointToPointDifference(subset0, subset1)
```

Similar to the previous pseudocode, `s` is an array of 256 bytes and consists of the S-box entries 0×63 to 0×16. The variable `trace` represents a data structure of all power traces captured during the attack. Individual access to each power trace and each point of the trace must be possible in this data structure. Finally, `subset0` and `subset1` are two arrays which will each store 128 individual power traces.

3.5. *Implementing Hamming Weight Power Model (CPA) attack*

Similar to Section 3.4, we wish to predict the output of the SubBytes step of AES. However, rather than sorting the output into two categories based on LSB, the goal of the Hamming Weight Power Model attack is to predict the number of bits set to 1 in the S-box output. The correlation coefficient can then be computed against the power trace and predicted power model against each key guess. The strongest linear correlation (negative or positive) should demonstrate the correct key guess.

The Hamming weight function is calculated by writing a method which will count the number of binary 1s of a given input (after converting the input data type to binary format). To implement the correlation coefficient, the open source .NET library named MathNet.Numerics [13] was used. A manual implementation of the correlation coefficient was initially used but it was found that MathNet.Numerics was better optimised, thus increased the speed of computation. The pseudocode which follows provides an example how one may implement this attack using a programming language of their own choosing.

```
for KeyGuess 0 to 255:
  empty hammingPower array
  reset pos1 to 1
  for plaintText 0 to 255:
        sBox_lookup = s[plainText XOR keyGuess]
        hammingPower[pos1] = getHammingWeight(sbox_lookup)
        increment pos1 by 1

  empty actualPower Array
  for p 0 to points:
        actualPower = getEachTracesDataPointAtPosition(p)
        calculatePearsonCoefficient(actualPower, hammingPower)
```

In the pseudocode presented above, `hammingPower` is an array which stores 256 individual hamming weight values while `actualPower` stores the 256 power consumption values as captured on the device under attack. Variable `points` is a count of how many data points each individual power trace contains. This value should be the same for all traces, so it is stored as a single integer. Finally, `pos1` is an integer used as a counter.

4. Results

Two sets of experiments are presented in this section: (1) DPA and CPA on 1-byte key and (2) DPA and CPA attacks on full 16-byte key of AES-128. The attack on a 1-byte key acts as a form of validation on the implementation of both DPA and CPA techniques while an attack on a full 16-byte key enables us to compare the viability of these two power analysis techniques in a side-by-side manner. The DPA attack implemented is referred to as the Difference of Means while the CPA attack involves modelling the power consumption of the device using the Hamming Weight Power Model technique.

For the 1-byte key attack, the key used is 65 while the 16-byte key used is 05, 08, 11, FE, 64, 09, 09, 54, 21, B1, B7, 23, 66, 14, 15, 7E. The graphs presented were created using the .NET C# software which was written to conduct each type of attack. Note that the labels for correct key guesses in the graphs were manually added to ease interpretation of results. In the graphs presented, the highest peak(s) in the Difference of Mean attack is our correct key guess while the lowest nadir(s) is the correct key guess in the Hamming Weight Power Model approach.

4.1. *Results of 1-byte attack on AES-128 using DPA and CPA*

Figures 9 and 10 present the result on the attack of a 1-byte key using Difference of Means and Hamming Weight Power Model, respectively.

4.2. *Results of 16-byte attack on AES-128 using DPA and CPA*

Figures 11 and 12 present the results on the attack of a 16-byte key using Difference of Means and Hamming Weight Power Model, respectively.

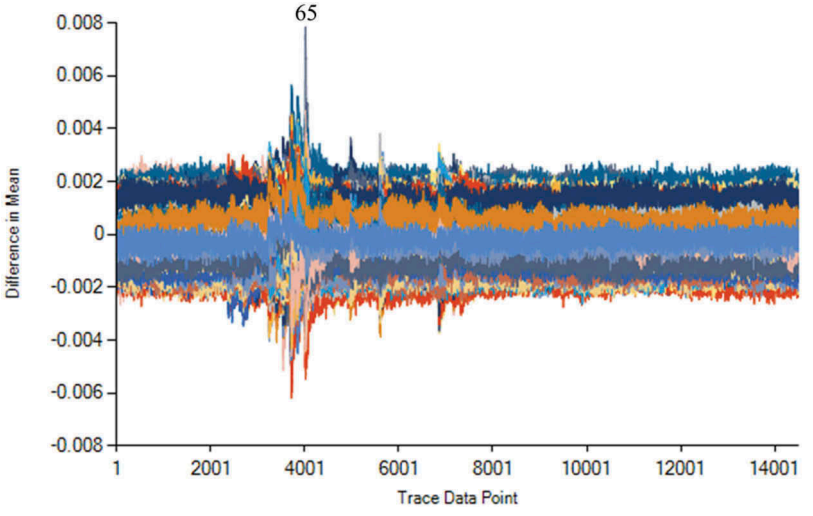

Figure 9. Difference of Means attack on 1-byte key.

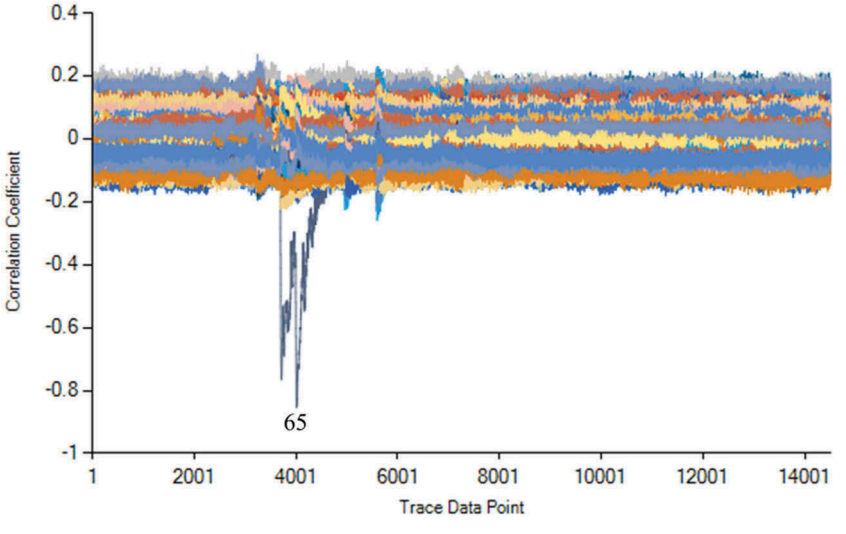

Figure 10. Hamming Weight Power Model attack on 1-byte key.

4.3. *Discussion*

Results from Section 4.1 demonstrate that an attack on a 1-byte key using both Difference of Means and Hamming Weight Power Model is both feasible. However, it is noted that the data leakage is far more distinct and discernible with the naked eye when using a CPA-based approach in comparison with the DPA approach. Although the Difference of Means attack shows that the highest peak is the correctly guess key, there is a greater degree of noise which potentially making results more difficult to interpret. The noise observed in the Difference of Mean result can be attributed to incorrect keys

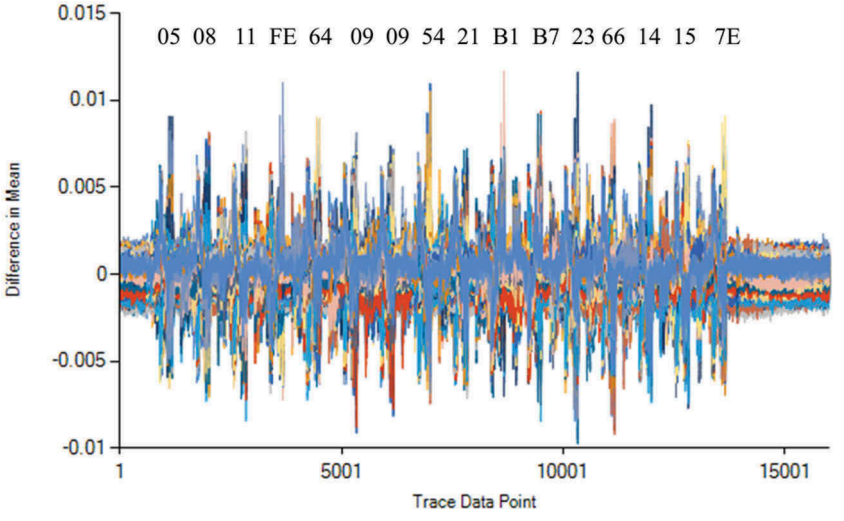

Figure 11. Difference of Means attack on 16-byte key.

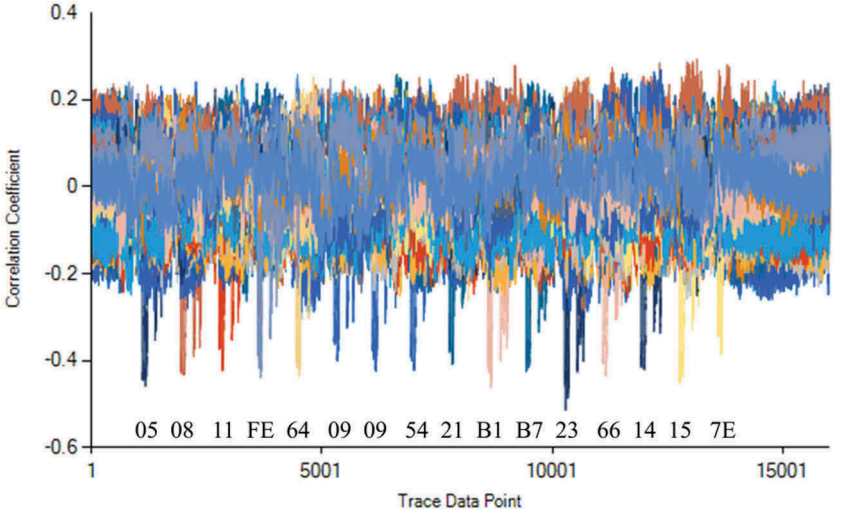

Figure 12. Hamming Weight Power Model attack on 16-byte key.

which produce partial correlation (i.e. there is a small but insignificant relationship between the incorrect key's differences of means relative to the selection function used). Noise observed in the Difference of Mean approach validates the observation originally made in [6] where the authors refer to this phenomenon as 'harmonics'.

Moving on to a full 16-byte key attack as shown in Section 4.2, both techniques were once again able to deduce the correct key values. However, as before, Difference of Means produces a far greater deal of noise in comparison with the Hamming Weight Power Model attack where 16 distinct nadirs may be observed. An observation of interest from the 16-byte attack is that since the SubBytes step of AES is conducted in a consecutive manner (i.e. the result of $P_1 \square K_1$ is used as a lookup the S-Box output then

P_2 K_2 and so on) the order of peaks (Difference of Means), and nadirs (Hamming Weight Power Model), of most significance relative to time represents the cipher key's order of input as well. Thus, even if we have two cipher keys of the same value, both keys will still be detected by the power analysis attack as the two cipher key values 09 demonstrates.

Overall, the time taken to gather the power traces for a full 16-byte key attack was approximately 25 minutes while the algorithm implemented to compute the DPA and CPA took no more than 5 minutes. Thus, the total time for each instance of this attack is around 30 minutes on average. From a practical perspective, we summarise that both forms of power analysis attacks are achievable on an Arduino Uno since the equipment and software detailed but, from an analytics perspective, the Hamming Weight Power Model attack proves to produce results which are more open to the ease of interpretation. Furthermore, the harmonic noise observed in the Difference of Means attack may prove more troublesome and, potentially, raise false positives if this methodology is applied to real-life devices and hardware. In Section 5, conclusion is given along with acknowledging some of the limitations of this work. Finally, some suggestions on future work is provided.

5. Related work

This work is built upon research originally proposed by Kocher et al. [5] where the fundamental idea behind DPA attacks was originally proposed. The paper by Kocher et al. also originally discussed the idea behind CPA attacks but formalised techniques were developed and expanded upon by numerous authors including Coron et al. [7], Brier et al. [14], Alioto et al. [15] and Mestiri et al. [8].

In the work by Kocher et al., attack on the AES algorithm using the Difference of Means technique was carried out on a Field-Programmable Gate Array (FPGA) board. The work of Coron et al. described the idea of using the Weight Power Model and experimentation was conducted on a smart card chip. Brier et al. expanded the work on CPA by proposing the use of a Hamming distance power model in place of Hamming weight while results were presented from data gathered from attacking an 8-bit chip against AES. Alioto et al. proposed a novel CPA attack named Leakage Power Attack which is derived from the theory behind the Hamming Weight Power Model and results were conducted against a MC74ACT273N chip. Lastly, Mestiri et al. describes an attack on a SASEBO-GII board against the AES algorithm with the goal of comparing the Hamming distance model against another derivative of the CPA called the Switching Distance model [16].

One of the core findings from these prior works is that after the publication by Kocher et al. focus has been given on development and comparison of CPA attacks. The popularity of CPA attacks may be explained by the work of Brier et al. where the authors have demonstrated that DPA attacks tend to produce ghost peaks, thus producing a greater number of false positives and increased difficulty in the analysis of results. The comparison made by Brier et al. was focused on comparison between DPA and their variant of CPA (Hamming distance) against the Data Encryption Standard (DES) algorithm rather than AES.

Our work differs in regard to previously cited authors in two respects: (1) focus is given on comparing the Difference of Means and Hamming Weight Power Model attack against the AES algorithm and (2) the device under test is an Arduino Uno. Although the results presented in this article agree with Brier et al.'s original finding that CPA attacks may produce cleaner results, we have provided quantifiable evidence by direct comparison of the two attack categories using the same device under test along with the same data gathering methodology. Additionally, despite evidence of ghost peaks, the results presented demonstrate that a Difference of Means attack is still applicable against the microcontroller under test.

We intentionally chose a device (the Arduino Uno) to perform an attack on which is now common and widely available. Disparity of experimental set-up was found in previous research since attacks were conducted against numerous different forms of chips and smart cards. Although there is nothing inherently wrong with this approach, some of the devices are now considerably dated, thus introducing difficulty in the acquisition of such hardware if one wishes to replicate results. Given the increasing popularity in microcontrollers (especially with the advances in IoT), providing a methodology and techniques on attacking cryptographic operations on an Arduino may be of great value for future researchers who wish to study this area of research.

Finally, on the subject of power analysis attacks against the Arduino Uno, we were able to find two works which were related to the topics of this paper. The first was written by Kang et al. [17] where they demonstrate an attack on the AddRoundKey step of AES using an Arduino Uno via the Hamming Weight Power Model. The second is an unpublished report by Banerjee et al. [18] that employs a similar technique to attacking AES running on an Arduino Uno with focus given to exploiting the SubBytes output. Our work here complements these two recent papers by expanding on the methodology required to mount such attacks on an Arduino Uno but is also differs since comparison and techniques are given to mounting both DPA and CPA attacks.

6. Conclusion, limitations and future work

This article aimed to perform attacks on the AES-128 algorithm using power analysis techniques. In particular, focus was given to comparison between two commonly discussed techniques: Difference of Means and Hamming Weight Power Model attack. The former attack may be categorised as a DPA attack while the latter is categorised as a CPA attack. The full methodology has been provided with pseudocode throughout this article in the hope of providing some value to researchers who wish to replicate and learn from these results.

We successfully demonstrate that both DPA and CPA techniques are viable in deducing the full 16-byte key of AES-128 by monitoring the power consumption of an Arduino Uno which implements the AddRoundKey and SubBytes steps in round 1 of AES. However, side-by-side comparison of the results produced by DPA and CPA demonstrate that the Hamming Weight Power Model approach may produce key guess results which are easier to interpret from an analytics perspective due to the lack of noise in comparison with the Difference of Means technique. Since results produced here have been gathered in a white-box testing environment, variation in findings may exist when applying these techniques real-life devices.

The core limitation in this work is how applicable the methodology is when attempts are made to perform attacks on real-life cryptographic devices running AES-128 under a black-box environment. We address two of the main challenges one may encounter when applying the methodology and techniques described in this article along with some suggestions for how they may be solved.

From observation of open source code of cryptographic libraries, one practice is to amalgamate all four steps (SubBytes, ShiftRows, MixColumns and AddRoundKey) in each round of encryption into a single iterative function. Thus, the output of SubBytes at Round 1 may not be accurately observed since no assignment of this variable may exist. Fortunately, this challenge could be solved by monitoring the power consumption during the final round of AES encryption where it is known that the last step will consist of a AddRoundKey function whereby the variable assignment must take place to output the result. By building an accurate selection function (DPA) or power model (CPA), one could predict the output of the final AddRoundKey step given known plaintext inputs. If prediction of output is correct, then one simply reverses each round of AES until they have the original starting cipher key values.

A second more challenging problem is of knowing when to capture power traces in a real-life device running cryptographic operations. This is a non-issue for the experiments conducted in this article since we may use pin 13 (LED) of the Arduino Uno to trigger the oscilloscope for capturing of traces but no such mechanism may exist in real-life devices. Further complexity is introduced since different devices may behave in a distinct manner depending on their hardware and software implementations. As a general suggestion to solve this problem, one should first attempt to identify the actual cryptographic algorithm running on the device. As Section 2.2 has shown, SPA alone may prove valuable in this case. Secondly, if it is identified that communication occurs between a real-life device and a PC under the attackers control during known cryptographic functions, it may be possible to use such responses as a capturing trigger. In the worst case scenario, it may still be possible to attack a device by gathering enough traces and ensuring they are aligned in a coherent manner.

Fundamentally, an attack on any real-life device running cryptographic operations may require some bespoke tuning of parameters. We hope the methodology demonstrated in this article provides a meaningful stepping stone in achieving such attacks while results presented in comparing DPA and CPA prove useful for researchers who wished to learn more about the theory and practical aspects of power analysis attacks.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by The Data Lab [Reg-15152].

Notes on contributors

Owen Lo received a BEng (2010) degree in Computer Networks and Distributed Systems from Edinburgh Napier University and also received a PhD (2015) degree in health informatics from the same institute. His current research is on side-channel attacks which is a technique used to attack a cryptographic system by monitoring the physical outputs of the system.

William J. Buchanan is Professor in the School of Computing at Edinburgh Napier University, and Fellow of the BCS and the IET. He currently leads The Cyber Academy (thecyberacademy.org) and the Centre for Distributed Computing, Networks, and Security, and works in the areas of security, data analysis, Cloud Security, Web-based infrastructures, e-Crime, cryptograph, triage, intrusion detection systems, digital forensics, mobile computing, agent-based systems, and security risk. Bill has one of the most extensive academic sites in the World (asecuritysite.com) and is involved in many areas of novel research and teaching in computing. He has published over 27 academic books and 250 academic research papers. Bill has many awards for excellence in knowledge transfer, innovation, and teaching, such as winning at the Edinburgh Napier University Student Excellence awards in 2011, 2014 and 2015. He also has an extensive track record for public engagement and social media and was included in the JISC Top 50 Higher Education Social Media Influencers in 2015. Bill also regularly appears on TV and radio, and was named as one of the top 100 people for Technology in Scotland for the last 2 years, and as one of the 50 people from Scotland's digital technologies industry changing the world (Times Scotland). In 2016, in Scotland, his work resulted in the Innovation of the Year related to advanced cryptography, and his research has led to three successful spin-out companies, including Zonefox (an innovative company focused on data loss detection), along with being awarded a number of patents.

Douglas Carson is Solutions Consultant working for Keysight Technologies' cyber solutions team in Edinburgh. During his career, he has architected measurement and processing solutions for leading edge telecoms technologies such as SS7 intelligent networks, voice over packet, 3G & 4G radio access networks and recently, network function virtualization. He is currently conducting research on cyber threat detection and forensic analysis using signal processing techniques in conjunction with Keysight test equipment. He holds nine patents in network protocol processing and correlation techniques.

ORCID

Owen Lo http://orcid.org/0000-0003-0201-6498

References

[1] Arduino. Arduino – ArduinoBoardUno [Internet]. 2016 [cited 2016 Sep 6]. Available from: https://www.arduino.cc/en/Main/ArduinoBoardUno

[2] Zhou Y, Feng D. Side-channel attacks: ten years after its publication and the impacts on cryptographic module security testing [Internet]. Cryptology ePrint archive. 2005 [cited 2016 Sep 6]. Available from: https://eprint.iacr.org/2005/388.pdf

[3] Kocher P. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. Adv Cryptol CRYPTO '96. 1996;1109:104–113.

[4] Boneh D, DeMillo RA, Lipton RJ. On the importance of checking cryptographic protocols for faults. In: Fumy W, editor. Advances in cryptology - EUROCRYPT '97: international conference on the theory and application of cryptographic techniques; May 11–15; Konstanz. Berlin: Springer; 1997. p. 37–51.

[5] Kocher P, Jaffe J, Jun B. Differential power analysis. In: Wiener M, editor. Advances in cryptology - CRYPTO' 99: 19th annual international cryptology conference; Aug 15–19; Santa Barbara (CA). Berlin: Springer; 1999. p. 388–397.
[6] Kocher P, Jaffe J, Jun B, et al. Introduction to differential power analysis. *J Cryptogr Eng*. 2011;1(1):5–27. DOI:10.1007/s13389-011-0006-y
[7] Coron J-S, Kocher P, Naccache D. Statistics and secret leakage. In: Frankel Y, editor. Financial cryptography: 4th international conference, FC 2000 Anguilla, British West Indies, February 20–24, 2000 proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg; 2001. p. 157–173.
[8] Mestiri H, Benhadjyoussef N, Machhout M, et al. A comparative study of power consumption models for CPA attack. *Int J Comput Netw Inf Secur*. 2012;5(3):25–31.
[9] Heron S. Advanced encryption standard (AES). *Netw Secur*. 2009;2009(12):8–12. DOI:10.1016/S1353-4858(10)70006-4
[10] Keysight Technologies. MSOX4104A mixed signal oscilloscope [Internet]. 2016 [cited 2016 Sep 6]. Available from: http://www.keysight.com/en/pdx-x201942-pn-MSOX4104A/mixed-signal-oscilloscope-1-ghz-4-analog-plus-16-digital-channels?cc=US=eng
[11] Keysight Technologies. Evaluating high-resolution oscilloscopes getting more than 8-bits of resolution from your 8-bit oscilloscope [Internet]. 2014 [cited 2016 Sep 6]. Available from: http://literature.cdn.keysight.com/litweb/pdf/5991-1617EN.pdf?id=2318232
[12] Pub NF. 197: advanced encryption standard (AES). *Fed Inf Process Stand Publ*. 2001;197:441–311.
[13] Math.Net. Math.Net numerics [Internet]. 2016 [cited 2016 Sep 6]. Available from: http://numerics.mathdotnet.com/
[14] Brier E, Clavier C, Olivier F. Correlation power analysis with a leakage model. *Cryptogr Hardw Embed Syst*. 2004;3156:16–29.
[15] Alioto M, Giancane L, Scotti G, et al. Leakage power analysis attacks: a novel class of attacks to nanometer cryptographic circuits. *IEEE Trans Circuits Syst I Regul Pap*. 2010;57(2):355–367. DOI:10.1109/TCSI.2009.2019411
[16] Peeters E, Standaert F-X, Quisquater -J-J. Power and electromagnetic analysis: improved model, consequences and comparisons. *Integr VLSI J*. 2007;40(1):52–60. DOI:10.1016/j.vlsi.2005.12.013
[17] Kang YJ, Kim TY, Jo JB, et al. An experimental CPA attack for Arduino cryptographic module and analysis in software-based CPA countermeasures. *Int J Secur Its Appl*. 2014;8(2):261–270.
[18] Banerjee U, Ho L, Koppula S. Power-based side-channel attack for AES key extraction on the ATMega328 microcontroller [Internet]. MIT CSAIL computer systems security group. 2015 [cited 2016 Sep 6]. Available from: http://css.csail.mit.edu/6.858/2015/projects/utsav-lisayz-skoppula.pdf