Differential Power Analysis attacks on AES

Kevin Meritt

Cryptography II

VCSG-706

May 18, 2012

INTRODUCTION

One of the most common block cipher algorithms used today is the Advanced Encryption Standard (AES) [2]. This algorithm has been studied for security against many different attack methods including brute force, linear and differential attacks. One very plausible attack method uses information that leaks through side channels [1]. These unintended leaks can be exploited by attackers and can be achieved in a straightforward manner quickly, and using relatively inexpensive equipment.

One of the most well-known and effective of the side-channel attacks today is that on information leaked through the power consumption. Differential Power Analysis (DPA) can extract a secret key by measuring the power used while a device is executing the AES algorithm.

SIDE CHANNEL ATTACKS

In a typical cryptosystem utilizing a block cipher, the cryptanalysis will typically start with a focus on the primary inputs and outputs as potential sources of information for an attack. These include the plaintext input, the secret key, and the resulting ciphertext output as shown in Figure 1.

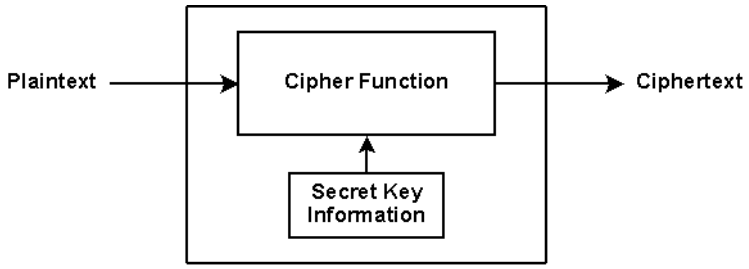

Figure 1 Ideal block cipher implementation inputs and outputs

Some assumptions that are made include that no secret key information is available to attackers and that the cryptographic function operates as a perfect black box. When these functions are implemented in real world devices however, these assumptions may not completely hold true. Other side channels exist that may leak information. If the information can be correlated to the secret key then these channels can be exploited by an attack.

There are several observable characteristics that can create side channels for potential information leakage. Some examples include power consumption, electromagnetic radiation, timing variations, and others [1] as shown in Figure 2.

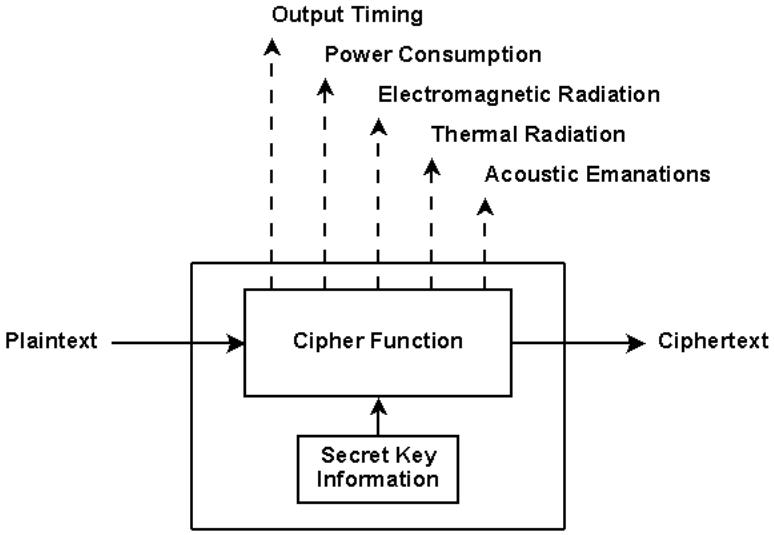

Figure 2 Indirect outputs from block cipher implementation

Side channel attacks have been developed that are able to successfully exploit the physical implementation that causes these leakages of information. For example, a cryptographic algorithm may differ in the execution time depending on the state of a bit in the key. This difference may be measurable and therefore utilized in a timing attack [1].

Power Analysis Attacks

One of the most powerful and common type today of side-channel attack is the power analysis attack. This attack utilizes information leaked by the power supply of a system during a cryptographic operation. Power analysis attacks attempt to find a relationship between the instantaneous power consumption and the internal state of a cryptographic implementation.

A power analysis attack can be essentially decomposed into the following three steps:

1. Identify a relationship between secret key information and the instantaneous power consumption. Also need to determine the required inputs to the system, the output values to be measured, and when to capture them.
2. Extract the state by measuring and recording the items identified earlier including the power consumption. The collection of measurements also known as traces can be made in a non-invasive manner while a system performs a cryptographic operation [5].
3. Evaluate the relationship between the measured items by processing the extracted information.

An important distinction in the relationship between secret key information and the instantaneous power consumption is the manner in which they are related. The two ways in which this occurs is at the algorithm level and the bit level.

At the algorithm level, execution of different operations tends to require different amounts of power. This difference is normally independent of the data values being manipulated. Therefore by examining the power trace, it may be possible to infer what operations are occurring. For example, Figure 3 shows the instantaneous power consumption of a cryptographic device as it performs the Data Encryption Standard (DES) algorithm [2]. The 16 individual rounds of the cipher are clearly visible from the repeating patterns. However the individual data bits being manipulated in the cipher cannot be visually determined.

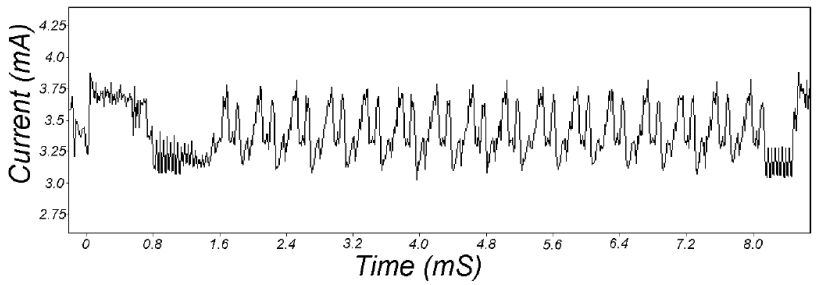

Figure 3 Power trace showing DES operation [1]

Despite not being able to directly ascertain data values, this form of analysis can be still be useful. It can be used to setup more powerful attacks by identifying the point within a cryptographic operation at which to attack.

Differences in instantaneous power consumption can also be related to the bit values that are being manipulated. As the bit values change, the underlying hardware consumes power, but on a much lower scale than the algorithm level [5]. Due to the subtle nature of the power variations, detection is more difficult. It may require modifications to the hardware and/or statistical techniques to identify and correlate the values. Figure 4 shows the variations in voltage, and therefore the power consumption, for a microprocessor accessing a memory location. As more data bit values are transitioning from logic 0 to logic 1, more power is consumed [8].

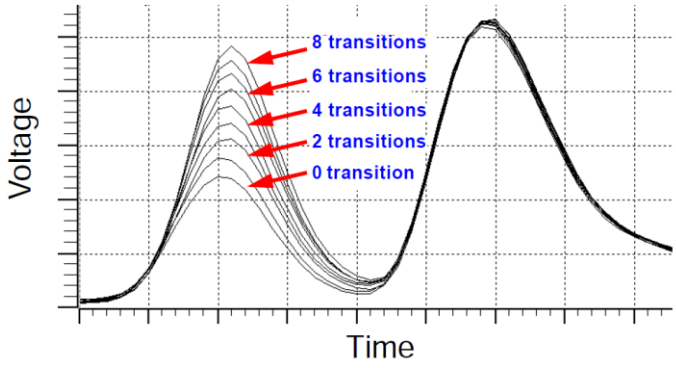

Figure 4 Variations in voltage due to bit transitions [8]

Unlike physical attacks, power analysis attacks are non-invasive, easily-automated, and can be mounted without knowing the design of the target device [5]. In addition, these attacks cannot generally be detected by a device, since the adversary's monitoring is normally passive.

Simple Power Analysis

The most basic form of power analysis is the Simple Power Analysis (SPA). This consists of a process of examining power traces for large scale differences which are caused by the operations being performed.

It has been shown that in some applications, it is possible to determine which software instructions are occurring or possibly which data bits are being changed [1]. In a worst case situation, it may be even be possible to determine the data bit values of secret information. Implementations where these differences are dependent on data values being manipulated are more vulnerable to an attack. This attack is very effective when an attacker has unlimited access to perform many encryption or decryption operations a device and also when the algorithm is known.

Differential Power Analysis

In most cases, SPA is insufficient alone to extract a secret key. Differential Power Analysis (DPA) is a much more powerful attack through the use of statistical analysis. The original concept of DPA [1] observed that there are power variations that can be correlated to data values that are being manipulated in a system. The variations are small and tend to be hidden by measurement errors and other forms of noise. However it is possible to overcome those by using statistical techniques tailored to a chosen target. These techniques involve the evaluation of the differences between the means of power traces in order to estimate a secret key. The traces are defined as a set of power consumption measurements taken from a system performing a cryptographic operation.

The basic steps to conduct a “difference of means” DPA test are as follows:

1. Perform many encryption (or decryption) operations on a cryptographic device with different sets of data.

2. Measure and record the power consumption and the data output (or input) processed during each operation.

3. Partition the power measurements into subsets according to a property of the state being processed.

4. Check for statistical differences between the subsets. When differences are observed, a data leak has been detected.

The algorithm details will now be presented for an attack with known values. Another possibility would be an attack with known plaintext values, depending on the specifics of the target system and which values were accessible to the attacker.

In order to implement a DPA attack, an attacker first observes *m* encryption operations and captures *T* power traces containing *k* samples each. This is represented as $T_{1:m}[1:k]$. The corresponding ciphertext values are also captured and represented as $C_{1:m}$. For this type of attack, knowledge of the plaintext is not required.

Once the power measurements and ciphertext values have been captured, the analysis phase is performed. The analysis uses the power consumption measurements to determine whether a key guess is correct. This is done by first establishing a selection function. A selection function $D(C, b, K_s)$ is defined that calculates the value of a target bit *b* given a ciphertext *C* and key guess K_s.

Given the $T_{1:m}[1:k]$ power traces and corresponding ciphertext values $C_{1:m}$ two groups of traces are constructed: one where $D(C, b, K_s) = 0$ and another where $D(C, b, K_s) = 1$. Next the attacker computes a *k*-sample differential trace $\Delta_D[j]$ by finding the difference between the average of traces for which $D(C, b, K_s) = 1$ and the average of traces for which $D(C, b, K_s) = 0$. Therefore $\Delta_D[j]$ is the average over $C_{1:m}$ of the effect due to the value represented by the selection function on the power consumption measurements at point *j*. The DPA algorithm is shown in Equation 1.

$$\begin{aligned} \Delta_D[j] &= \frac{\sum_{i=1}^{m} D(C_i, b, K_s)\mathbf{T}_i[j]}{\sum_{i=1}^{m} D(C_i, b, K_s)} - \frac{\sum_{i=1}^{m} (1 - D(C_i, b, K_s))\,\mathbf{T}_i[j]}{\sum_{i=1}^{m} (1 - D(C_i, b, K_s))} \\ &\approx 2\left(\frac{\sum_{i=1}^{m} D(C_i, b, K_s)\mathbf{T}_i[j]}{\sum_{i=1}^{m} D(C_i, b, K_s)} - \frac{\sum_{i=1}^{m} \mathbf{T}_i[j]}{m}\right) \end{aligned} \quad (1)$$

If the key guess K_s is correct, the average trace for $D(C, b, K_s) = 1$ will be slightly higher at the point of correlation and the average trace for $D(C, b, K_s) = 0$ will be slightly lower. Therefore the bit computed by the selection function $D(C, b, K_s)$ will equal the actual value of target bit *b* with probability P = 1. It is effectively correlated in this case to what was actually computed by the target device.

If the key guess K_s is incorrect, the bit computed by the selection function $D(C, b, K_s)$ will equal the correct bit value with probability P = ½ or half the ciphertexts. It is effectively uncorrelated in this case to what was actually computed by the target device.

So if K_s is incorrect then: $\lim_{m \to \infty} \Delta_D[j] \approx 0$

The correct value of K_s will be identified on a plot by the spikes in the differential trace as shown in Figure 5.

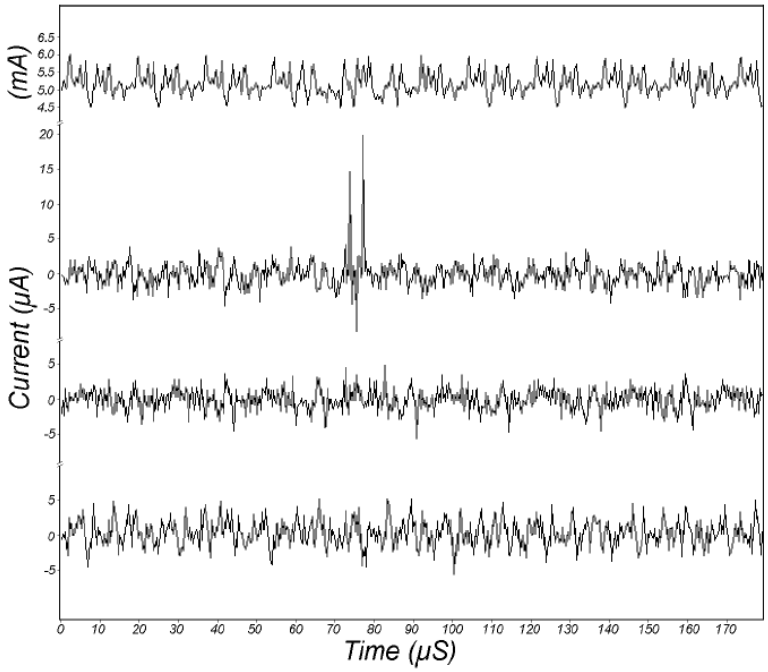

Figure 5 DPA traces [1]

Note the top trace shows the average power consumption value. The next three are differential traces where the first was produced with a correct guess for K_s. The lower two traces were produced using incorrect values for K_s.

Correlation Power Analysis

Correlation Power Analysis (CPA) is an extension of DPA where a model of the power consumption is created for use in the analysis phase of an attack. The model needs to approximate the power consumption of the target cryptographic device during an encryption operation. The resulting power predicted by the model will then be correlated to the actual measured power consumption using a key hypothesis. The highest peak of the correlation plot gives the correct key hypothesis.

There are multiple methods for constructing this model, including simulation in an environment designed to show the power consumption [14]. If the architecture of the target device is known, then an accurate prediction of the power consumption can be made. In cases where the architecture is not entirely known or it is not possible to simulate, a more general power model can be used. Two common power models in use today are the Hamming weight and Hamming distance models [12].

The Hamming weight model is a basic power model which is based on the assumption that the amount of power consumed is proportional to the number of bits that are logic '1' during an operation. The greater the number of bits that are set will result in a larger amount of power consumed. This model can be used when little information is known about the architecture of a target device [12].

The Hamming distance power model is an extension of the Hamming weight model, which uses the number of logic transitions during a cryptographic operation to predict the power usage. If a bit is static during an operation, then it is assumed that it will not contribute to the power. Another assumption is that '0' to '1' and '1' to 0' transitions consume the same amount of power. This power model is generally used for Correlation Power Analysis.

A commonly used measure of correlation is the Pearson Product-Moment Correlation which is also called Pearson's correlation coefficient [11]. Pearson's correlation coefficient value ρ (rho) reflects the degree of linear relationship between two variables X and Y as represented by Equation 2.

$$\rho_{X,Y} = \operatorname{corr}(X,Y) = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} \qquad (2)$$

The covariance (cov) is a measure of how much two random variables change together. This is divided by the standard deviations of X and Y represented by σ_X and σ_Y. The coefficient value ranges from +1 to -1. A computed correlation of +1 indicates that there is a perfect positive linear relationship between variables while a value of -1 indicates there is a perfect negative linear relationship. If the correlation equals 0 then there is no linear relationship between the two variables.

When there is a series of *n* measurements of *X* and *Y* written as x_i and y_i for *i* = 1 to n, then the Pearson correlation can be estimated by the sample correlation coefficient r_{xy}, shown in Equation 3 where x-bar and y-bar are the sample means of X and Y, and s_x and s_y are the sample standard deviations of X and Y [30].

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y} \qquad (3)$$

For CPA, the variables X and Y will be used to represent the correlation between the power consumption measurement samples and power consumption hypothesis samples. The hypothesis samples are created from the power model, which is typically based on the Hamming distance power model [30].

Advanced Encryption Standard

The Advanced Encryption Standard (AES) is a cryptographic block cipher algorithm in widespread use today. Adopted as an official standard in 2001 as the "Federal Information Processing Standards Publication 197", AES is an encryption algorithm that is commonly used for encryption and decryption of secure digital data [2].

The AES encryption algorithm is a symmetric key block cipher that takes 128 bits as input called plaintext, and outputs 128 bits encrypted as ciphertext. The data format is arranged into a 4x4 matrix called the State as shown in Figure 6. Each column of the matrix is comprised of 4 bytes. The cipher key used to encrypt and or decrypt the data can be 128, 192, or 256 bits in length with the strength of the

encryption increasing with the length of the cipher key. For the remainder of this document, the focus will be on the 128 bit cipher key version, referred to as AES-128.

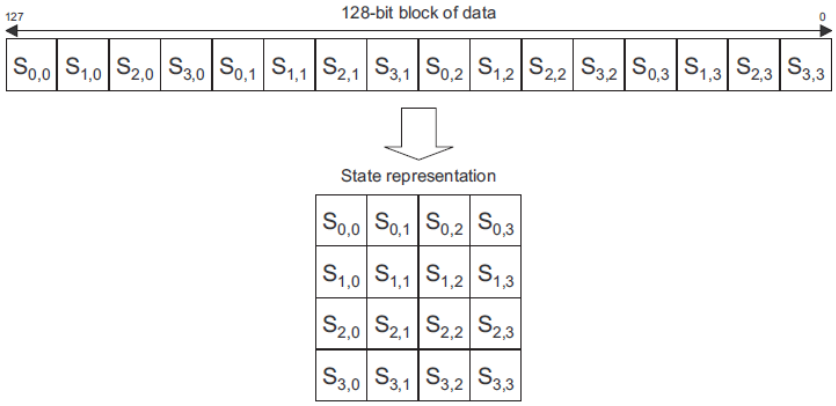

Figure 6 State representation of 128-bit data blocks

The AES encryption algorithm is broken up into four transforms known as SubBytes, ShiftRows, MixColumns and AddRoundKey. The decryption algorithm uses the inverse transforms in reverse order to recover the original plaintext. There is also a Key Expansion that must be performed on the cipher key. The individual transforms are performed in a number of rounds that are dependent on the cipher key size. For key sizes of 128, 192, or 256 bits there are a total of 10, 12, or 14 rounds respectively. A top level view of transformation flow from plaintext to ciphertext is shown in Figure 7.

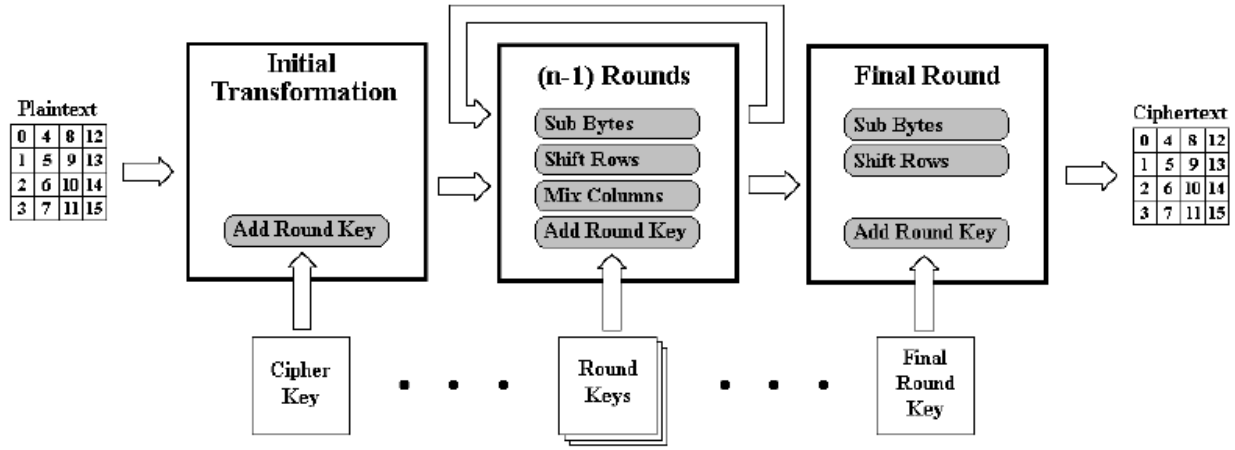

Figure 7 AES Algorithm [29]

In order to implement the AES standard, the algorithm requires the arithmetic to be performed in the finite field GF(2^8). Therefore addition and subtract are performed modulo 2. Also since all byte values are interpreted as finite field elements, the bits in the state represent the coefficients of the polynomials. Multiplication is performed with conventional polynomials except modulo an irreducible polynomial of degree 8. The irreducible polynomial chosen is $m(x) = x^8 + x^4 + x^3 + x + 1$.

The SubBytes transformation is a nonlinear transform that replaces each byte in the state with a byte from an invertible substitution table (S-box). SubBytes is comprised of a multiplicative inverse in the finite field GF(2^8) and an affine transformation over GF(2). This step can be computed dynamically or by utilizing a look-up table as shown in Figure 8.

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

Figure 8 AES S-box substitution values [2]

The ShiftRows transformation cyclically shifts the rows of the state to provide horizontal diffusion. The first row is not shifted, the second row left shifted one byte, the third row is left shifted two bytes and the fourth row is left shifted three bytes. The bytes are shown in Figure 9.

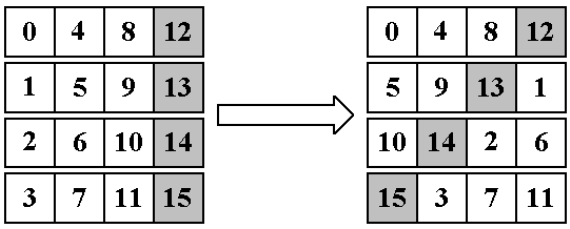

Figure 9 AES ShiftRows operation [29]

The MixColumns transformation treats each word as a four term polynomial in a Galois Field GF(2^8) to provide vertical diffusion. Each polynomial is multiplied modulo $x^4 + 1$ with a fixed polynomial ($3x^3+x^2+x+2$). This is the most processing intensive transformation in AES since it involves a multiply operation. Figure 10 shows this step as a matrix multiplication. The inverse MixColumns transform (invMixColumns) used for decryption multiplies the encrypted ciphertext by $a^{-1}(x)$ in the decryption algorithm.

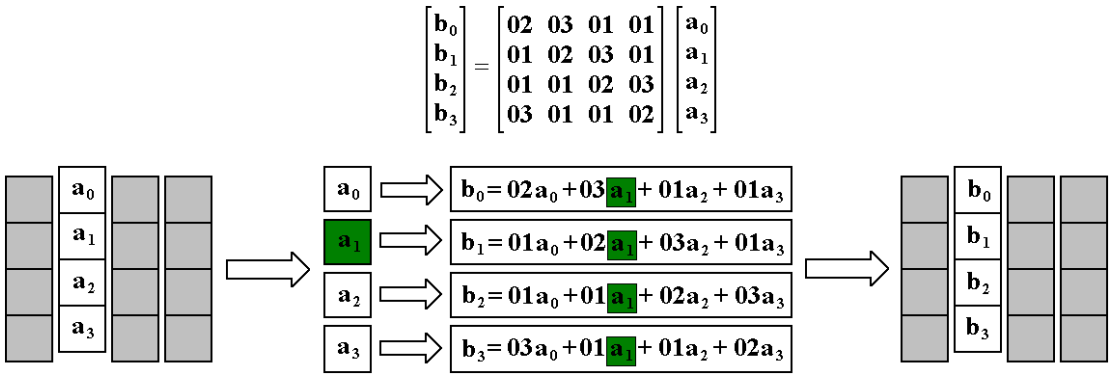

Figure 10 AES MixColumns operation [29]

The AddRoundKey transform is a simple bitwise addition with a key generated from the Key Expansion algorithm. Since Galois Field addition of binary numbers is an XOR operation for both addition and subtraction, the AddRoundKey function does not need an explicit inverse function for decryption.

The cipher key is expanded into round keys according to the key schedule. One column of the round key is generated at a time. Most columns are generated by combining the bytes of the preceding column with the column 4 positions to the left. However the shaded columns are computed by first transforming the previous column with the operation T(x) shown in Figure 11.

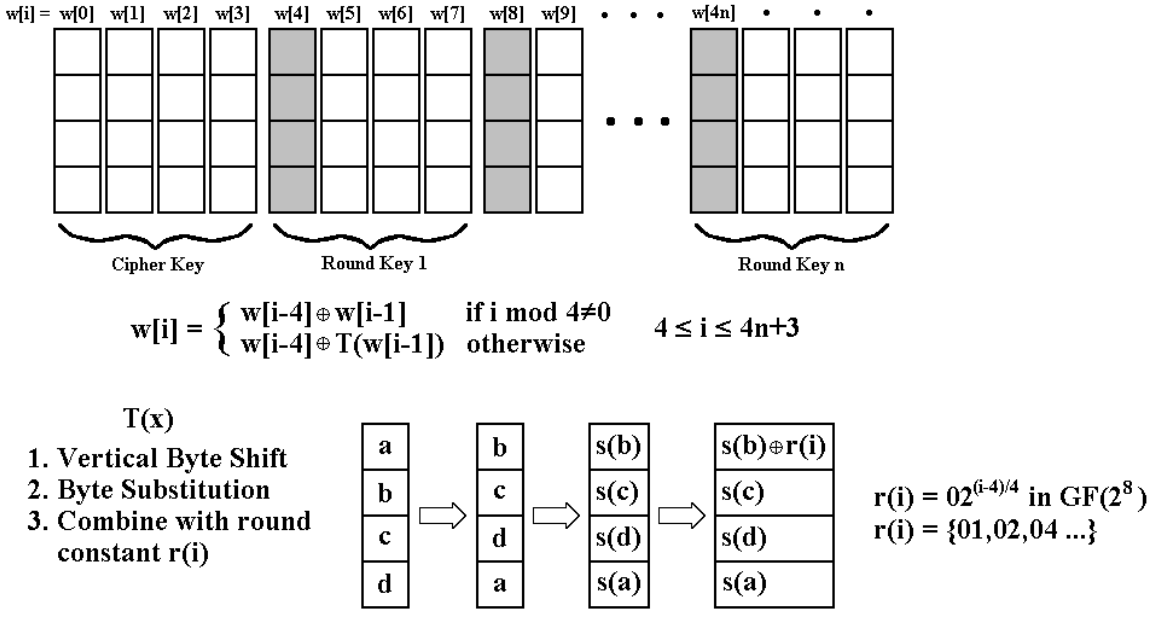

Figure 11 AES key expansion routine [29]

Pseudo code for the key expansion algorithm is provided in Figure 12 [2] so that intermediate values from an example can be referenced later in Figure 13.

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
   word  temp

   i = 0

   while (i < Nk)
      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
      i = i+1
   end while

   i = Nk

   while (i < Nb * (Nr+1)]
      temp = w[i-1]
      if (i mod Nk = 0)
         temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
      else if (Nk > 6 and i mod Nk = 4)
         temp = SubWord(temp)
      end if
      w[i] = w[i-Nk] xor temp
      i = i + 1
   end while
end
```

Figure 12 Pseudo code for AES key expansion [2]

The RotWord function performs a cyclic permutation of a word by doing a one-byte left circular operation. The Rcon function sets the right three bytes to zero. The remaining byte becomes 1 for the first round and two times the value from the previous round.

Figure 13 shows the expansion for a 128-bit cipher key. The key schedule result is only shown for rounds 1 and 10 for conciseness. The complete schedule can be found in Appendix A.1 of [2]. The initial cipher key hex value is "2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c". So for $N_k = 4$, the initial word values are: w_0 = 2b7e1516 w_1 = 28aed2a6 w_2 = abf71588 w_3 = 09cf4f3c. The computed final round key value is: "d0 14 f9 a8 c9 ee 25 89 e1 3f 0c c8 b6 63 0c a6".

i (dec)	temp	After RotWord()	After SubWord()	Rcon[i/Nk]	After XOR with Rcon	w[i-Nk]	w[i]= temp XOR w[i-Nk]
4	09cf4f3c	cf4f3c09	8a84eb01	01000000	8b84eb01	2b7e1516	a0fafe17
5	a0fafe17					28aed2a6	88542cb1
6	88542cb1					abf71588	23a33939
7	23a33939					09cf4f3c	2a6c7605
8	2a6c7605	6c76052a	50386be5	02000000	52386be5	a0fafe17	f2c295f2
9	f2c295f2					88542cb1	7a96b943
10	7a96b943					23a33939	5935807a
11	5935807a					2a6c7605	7359f67f
12	7359f67f	59f67f73	cb42d28f	04000000	cf42d28f	f2c295f2	3d80477d

40	575c006e	5c006e57	4a639f5b	36000000	7c639f5b	ac7766f3	d014f9a8
41	d014f9a8					19fadc21	c9ee2589
42	c9ee2589					28d12941	e13f0cc8
43	e13f0cc8					575c006e	b6630ca6

Figure 13 AES Key expansion example [2]

An example cipher from Appendix B from [2] will demonstrate the transformation of the state for the AES algorithm. Figure 14 shows the values in the state array as the cipher progresses. They are only shown for rounds 1 and 10 for conciseness. The example given uses a plaintext input hex value of "32 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34" and a cipher key value of "2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c". Note the example uses the same cipher key value from above. The resulting ciphertext value after the final round is "39 25 84 1d 02 dc 09 fb dc 11 85 97 19 6a 0b 32".

Round Number	Start of Round	After SubBytes	After ShiftRows	After MixColumns		Round Key Value	
input	32 88 31 e0 43 5a 31 37 f6 30 98 07 a8 8d a2 34				⊕	2b 28 ab 09 7e ae f7 cf 15 d2 15 4f 16 a6 88 3c	=
1	19 a0 9a e9 3d f4 c6 f8 e3 e2 8d 48 be 2b 2a 08	d4 e0 b8 1e 27 bf b4 41 11 98 5d 52 ae f1 e5 30	d4 e0 b8 1e bf b4 41 27 5d 52 11 98 30 ae f1 e5	04 e0 48 28 66 cb f8 06 81 19 d3 26 e5 9a 7a 4c	⊕	a0 88 23 2a fa 54 a3 6c fe 2c 39 76 17 b1 39 05	=
10	eb 59 8b 1b 40 2e a1 c3 f2 38 13 42 1e 84 e7 d2	e9 cb 3d af 09 31 32 2e 89 07 7d 2c 72 5f 94 b5	e9 cb 3d af 31 32 2e 09 7d 2c 89 07 b5 72 5f 94		⊕	d0 c9 e1 b6 14 ee 3f 63 f9 25 0c 0c a8 89 c8 a6	=
output	39 02 dc 19 25 dc 11 6a 84 09 85 0b 1d fb 97 32						

Figure 14 AES algorithm intermediate state values

Differential Power Analysis on AES

There are certain properties that can be seen in the state transformation of Figure 14 that could be exploited to find a correlation in a power analysis attack. The first is that the initial plaintext input is XORed with the initial cipher key. Since there is a direct relationship between the two values, data bits at that operation could be used as a potential target for an attack. This would require that the plaintext values were accessible.

The second property occurs in the final round. The final ciphertext output is a result of an intermediate value XORed with the final round key value. This also has the property of not having the MixColumns vertical diffusion included for the last round. The relationship between state and key is not quite as direct as the previous, but data bits at that operation could be exploited in an attack. If an attack was able to successfully determine the final round key value, the initial key could be computed by reversing the key expansion operation. This property as well as the previous is shown in Figure 15.

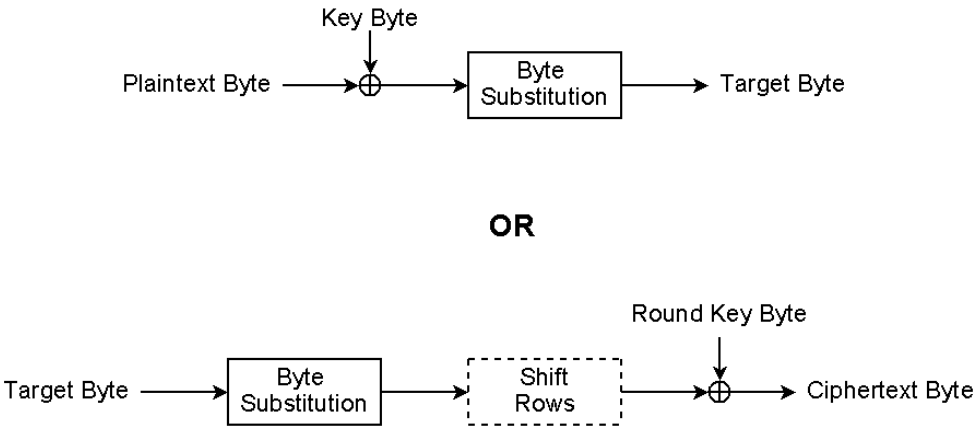

Figure 15 DPA target on AES [29]

Another AES algorithm property that could be exploited is that the S-box substitution used in the SubBytes step operates on each byte independently. Therefore the attack can be performed on one selected byte at a time and then repeated 16 times for the remaining bytes, using the same set of power traces to obtain the full 128 bit key. This significantly decreases the magnitude of the attack space. Instead of having to do an exhaustive key search of 2^{128} possibilities, the problem becomes one of only 16 x 28 possible combinations.

Using the relationships established above between the state and the cipher key, a DPA attack specific to the AES-128 algorithm can now be described. For a known ciphertext DPA attack, the assumption is that the output ciphertext values are known, but the plaintext and cipher key are not. The attack will choose an intermediate bit to analyze at a target byte location, such as the SubBytes byte substitution operation shown in Figure 15. Since the focus is on the final round, power consumption measurements will be made around that point in time while the device is performing the encryption of random, unknown plaintext values. The power consumption associated with each ciphertext value will be collected and recorded for post processing.

After collecting many values, the next step is the partitioning and analysis. One byte of the key (associated with the selected intermediate position) will be chosen and a guess will be made for its value (256 possible choices). Next the selected bit position chosen earlier will be calculated by inverting the SubBytes operation and then XORed with the partial key guess, as shown in Equation 4. The inversion is obtained by applying the inverse of the affine transformation followed by taking the multiplicative inverse in GF(2^8) [2]. This can be done utilizing a look-up table similar to Figure 8 shown earlier.

$$D = S^{-1}(C_i \text{ XOR } K_i) \qquad (4)$$

Next the power traces will be partitioned into two subsets based on the computed values for *D*. Traces for *D* = 0 go into one subset and *D* = 1 into the other. Next the average of the two subsets will be calculated. Then the differences between the averages are examined.

When the average depends on the selected input bit, there will be a correlation or spike in the data as shown in Figure 16. Otherwise the data will trend to zero, indicating no correlation, or simple noise, as in Figure 17.

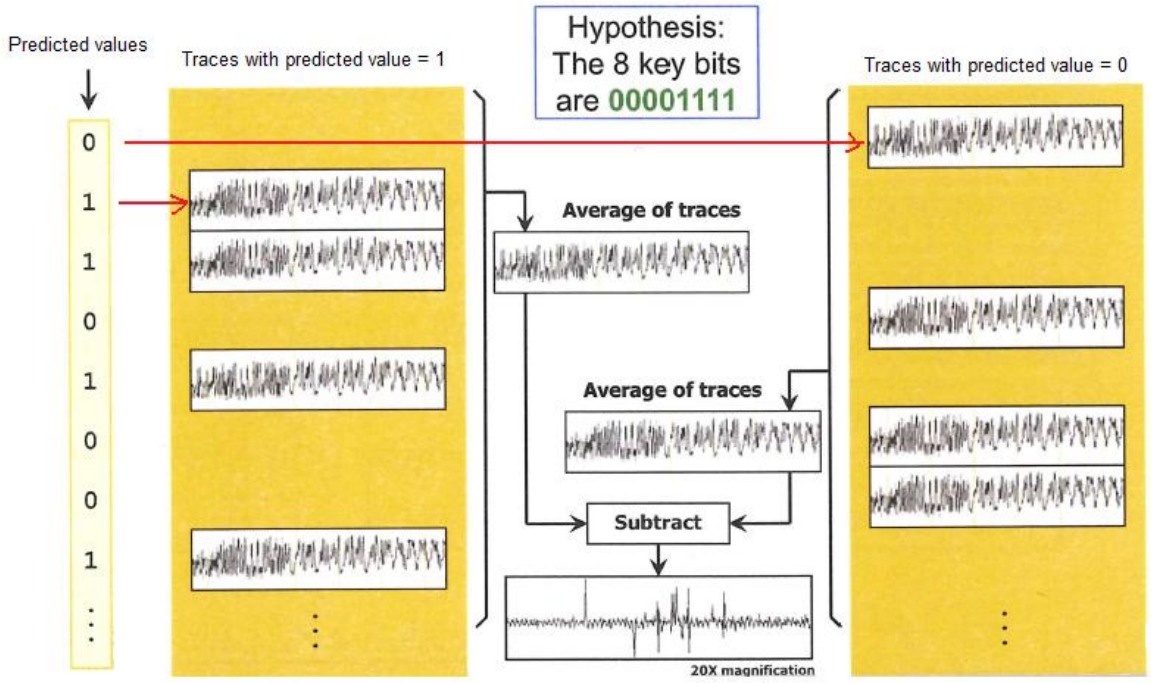

Figure 16 DPA with correct Key guess [20]

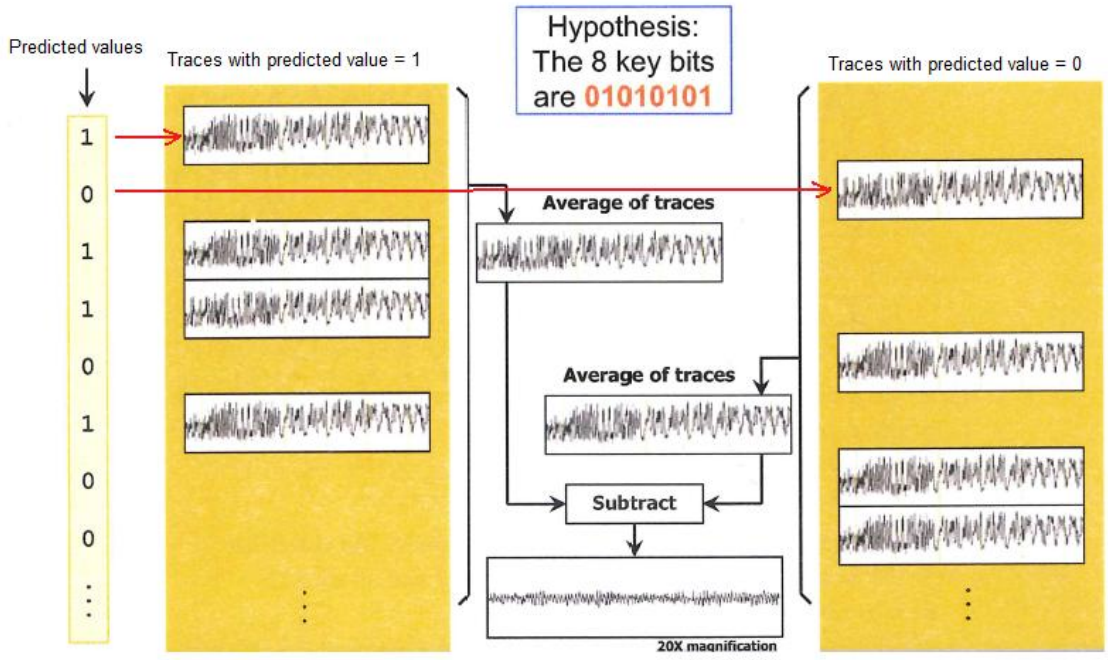

Figure 17 DPA with incorrect Key guess [20]

This process will then be repeated for the remaining key bytes, but using the same measurements already collected. The number of power traces required to extract the full 128-bit key is typically in the thousands, depending on the quality of signal measurements [6][11][12].

Correlation Power Analysis on AES

For a CPA attack on an AES implementation, the first step is to analyze the design and select a particular sensitive data register for the attack. The data in the register must hold a relationship with power consumption in order to create a valid power model. As noted earlier for the AES algorithm, a relationship exists between the final ciphertext output and the intermediate S-box value during the final round of encryption. This is a likely target location to attack. A typical hardware implementation of AES showing the transformation blocks is shown in Figure 18. The register of interest is labeled RB located in the data path prior to the SubBytes transformation and ciphertext output.

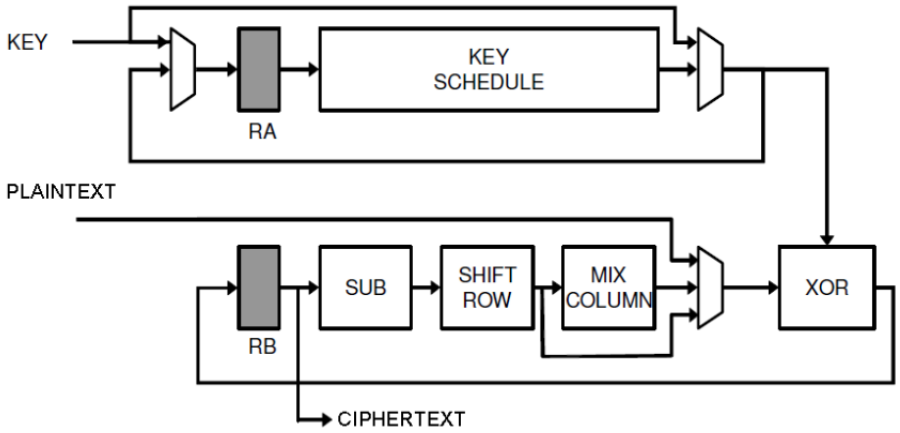

Figure 18 AES implementation [11]

Once the sensitive data location is selected, a power model can be determined. In this case, the Hamming distance of the data transition of an 8-bit register block at the S-box boundary.

The next step for CPA is similar to DPA in that many encryption operations are performed using random plaintext values and an unknown cipher key. Then the output ciphertext and associated power trace is measured and recorded. Then for each set, an 8-bit partial key value is guessed for the final encryption round. Next, the Hamming distance of the data transition of each 8-bit register block at the S-box boundary is calculated for each ciphertext value. Power traces are then sorted into groups associated with the calculated Hamming values.

Next, the Pearson's sample correlation coefficient equation can be utilized to determine the correlation between the power and the sensitive data. Using Equation 3, x_i represents the measured power samples and y_i represents the calculated power model samples from the Hamming distance model. If a correlation occurs then there will be a spike in the graph for the correct key byte value as shown in Figure 19.

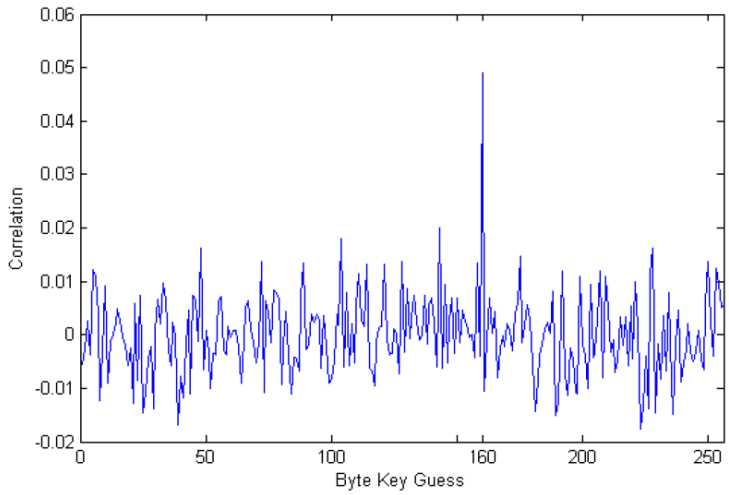

Figure 19 AES Correlation [30]

BIBLIOGRAPHY

[1] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” proceedings of CRYPTO ’99, *Lecture Notes in Computer Science*, vol. 1666, Springer, pp. 388–397, 1999.

[2] National Institute of Standards and Technology (NIST) of U.S. Department of Commerce, “FIPS 197: Advanced Encryption Standard,” Nov. 2001.

[5] F.-X. Standaert, “Introduction to Side-Channel Attacks,” in *Secure Integrated Circuits and Systems*, pp. 27–44, Springer, 2009.

[6] T. Katashita, A. Satoh, K. Kikuchi, N. Nakagawa and M. Aoyagi, “DPA Characteristic Evaluation of SASEBO for Board Level Simulations,” proceedings of COSADE 2010, pp.36-39, 2010.

[8] K. Tiri, M. Akmal, I. Verbauwhede. “A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards,” in Proc. Eur. Solid-State Circuits Conf. (ESSCIRC), Florence, Italy, 2002, pp. 403–406.

[11] S. Shah, R. Velegalati, J. Kaps, D. Hwang, “Investigation of DPA Resistance of Block RAMs in Cryptographic Implementations on FPGAs,” *International Conference on Reconfigurable Computing and FPGAs (ReConFig) 2010*, pp.274-279, Dec. 2010.

[12] R. Velegalati, J. Kaps, “DPA resistance for light-weight implementations of cryptographic algorithms on FPGAs,” *International Conference on Field Programmable Logic and Applications 2009*. FPL 2009, pp.385-390, 2009.

[14] K. Tiri and I. Verbauwhede, “A Logic level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation,” Proceedings of DATE2004, pp. 246–251, 2004.

[20] CRI, “DPA Workstation Training: Differential Power Analysis,” Cryptographic Research Inc., Jan. 2012.

[29] K. Smith Jr., “Methodologies for power analysis attacks on hardware implementations of AES,” Master's thesis, Rochester Institute of Technology, 2009.

[30] W. Hnath, J. Pettengill, “Differential Power Analysis Side-Channel Attacks in Cryptography,” Major Qualifying Project, Worcester Polytechnic Institute, April 2010.