Improved “Partial Sums”-based Square Attack on AES

Michael Tunstall

Department of Computer Science, University of Bristol, Merchant Venturers Building, Woodland Road, Bristol, U.K.

Keywords: Cryptanalysis, Square Attack, Advanced Encryption Standard.

Abstract: The Square attack as a means of attacking reduced round variants of AES was described in the initial description of the Rijndael block cipher. This attack can be applied to AES, with a relatively small number of chosen plaintext-ciphertext pairs, reduced to less than six rounds in the case of AES-128 and seven rounds otherwise and several extensions to this attack have been described in the literature. In this paper we describe new variants of these attacks that have a smaller time complexity than those present in the literature. Specifically, we demonstrate that the quantity of chosen plaintext-ciphertext pairs can be halved producing the same reduction in the time complexity. We also demonstrate that the time complexity can be halved again for attacks applied to AES-128 and reduced by a smaller factor for attacks applied to AES-192. This is achieved by eliminating hypotheses on-the-fly when bytes in consecutive subkeys are related because of the key schedule.

1 INTRODUCTION

The Advanced Encryption Standard (AES) (FIPS PUB 197, 2001) was standardized in 2001 from a proposal by Daemen and Rijmen (Daemen and Rijmen, 1998). It has since been analyzed with regard to numerous attacks ranging from purely theoretical cryptanalysis to attacks that require some extra information, e.g from some side channel (Mangard et al., 2007), to succeed.

In Daemen and Rijmen’s AES proposal an attack is described that is referred to as the Square attack (Daemen and Rijmen, 1998). This attack was so-called since it was first presented in the description of the block cipher Square (Daemen et al., 1997). The Square attack is based on a particular property arising from the structure of AES. That is, for a set of 256 plaintexts where each byte at an arbitrary index is a distinct value and all the other bytes are equal, the XOR sum of the 256 intermediate states after three rounds of AES will be zero.

Some optimizations to this attack have been proposed in the literature. Ferguson et al. proposed a way of conducting the Square attack referred to as the “partial sums” method (Ferguson et al., 2001). This allowed the Square attack to be conducted with a relatively low time complexity for reduced round variants of AES. The time complexity of these attacks was further reduced following an observation made by Lucks. He noted that given a known last subkey in AES-192 then information on previous subkeys can be derived.

Recent cryptanalytical attacks have predominantly been focused on other properties, such as impossible differentials (Bahrak and Aref, 2008; Lu et al., 2008; Zhang et al., 2007). The use of impossible differentials is related to the Square attack but allows an attacker to overcome variants of AES with more rounds. Recently, a marginal attack on AES has also been proposed that is based on the use of bicliques (Bogdanov et al., 2011). However, for suitably reduced round variants of AES the “partial sums” method proposed by Ferguson et al. is currently the most efficient chosen plaintext attack.

In this paper we describe how the attacks proposed by Ferguson et al. and Lucks can be improved. Specifically, we show that the number of chosen plaintext-ciphertext pairs required to conduct the Square attack can be halved and therefore halve the time complexity of the attack. Moreover, we demonstrate that the time complexity of the Square attack can be halved again when applied to AES-128, and reduced to a lesser extent for AES-192, by exploiting relationships between key bytes as they are derived. In this paper we restrict ourselves to attacks that require a relatively small number of chosen plaintext-ciphertext pairs. The attacks proposed by Ferguson et al., based on the Square attack, that require around 2^{128} chosen plaintext-ciphertext pairs are beyond the cope of this paper (Ferguson et al., 2001).

Tunstall M..
Improved “Partial Sums”-based Square Attack on AES.
DOI: 10.5220/0003990300250034
In *Proceedings of the International Conference on Security and Cryptography* (SECRYPT-2012), pages 25-34
ISBN: 978-989-8565-24-2

This paper is organized as follows. In Section 2 we define the notation we use to describe AES. In Section 3 we describe the property that the Square attack is based on. In Section 4 we describe how the Square attack can be applied to AES-128, and in Section 5 we describe how the Square can be applied to AES-192 and AES-256. We summarize our contribution and conclude in Section 6.

2 PRELIMINARIES

In this paper, multiplications in $\mathbb{F}_{2^8}$ are considered to be polynomial multiplications modulo the irreducible polynomial $x^8+x^4+x^3+x+1$. It should be clear from the context when a mathematical expression contains integer multiplication.

2.1 The Advanced Encryption Standard

Algorithm 1: The AES encryption function.

```
Input: The 128-bit plaintext block $P$ and 128,
       192 or 256-bit secret key $K$, with $N$ set
       to 10, 12, 14 respectively.
Output: The 128-bit ciphertext block $C$.
$X \leftarrow$ AddRoundKey$(P,K)$ ;
for $i \leftarrow 1$ to $N$ do
    $X \leftarrow$ SubBytes$(X)$ ;
    $X \leftarrow$ ShiftRows$(X)$ ;
    if $i \neq N$ then
        $X \leftarrow$ MixColumns$(X)$ ;
    end
    $X \leftarrow$ AddRoundKey$(X,K)$ ;
end
$C \leftarrow X$ ;
return $C$
```

The structure of the Advanced Encryption Standard (AES) , as used to perform encryption, is illustrated in Algorithm 1. In discussing the AES we consider that all intermediate variables of the encryption operation variables are arranged in a 4×4 array of bytes, referred to as the state matrix. For example, the 128-bit plaintext $P = (p_1, p_2, \ldots, p_{16})$, where each $p_i \in \{1, \ldots, 16\}$ is one byte, is arranged in the following fashion

$$\begin{pmatrix} p_1 & p_5 & p_9 & p_{13} \\ p_2 & p_6 & p_{10} & p_{14} \\ p_3 & p_7 & p_{11} & p_{15} \\ p_4 & p_8 & p_{12} & p_{16} \end{pmatrix} .$$

The encryption itself is conducted by the repeated use of a round function that comprises the following operations executed in sequence:

The AddRoundKey operation XORs each byte of the array with a byte from a corresponding subkey. In each instance of the AddRoundKey a fresh 16-byte subkey is used from the subkey bytes generated by the key schedule. We describe how this is done in more detail for the different variants of AES in Section 2.2.

The SubBytes operation is the only nonlinear step of the block cipher, consisting of a substitution table applied to each byte of the state. This replaces each byte of the state matrix by its multiplicative inverse, followed by an affine mapping. In the remainder of this paper we will refer to the function S as this substitution table and S^{-1} as its inverse.

The ShiftRows operation is a byte-wise permutation of the state that operates on each row.

The MixColumns operation operates on the state column by column. Each column of the state matrix is considered as a vector where each of its four elements belong to $\mathbb{F}_{2^8}$. A 4×4 matrix M whose elements are also in $\mathbb{F}_{2^8}$ is used to map this column into a new vector. This operation is applied to the four columns of the state matrix. Here M and its inverse M^{-1} are defined as

$$M = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \text{ and} M^{-1} = \begin{pmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{pmatrix}$$

All the elements in M and M^{-1} are elements of $\mathbb{F}_{2^8}$ expressed in decimal.

In Algorithm 1 we can see that the last round does not include the execution of a MixColumns operation. In all the attacks considered in this paper we will assume that the last round does not include a MixColumns operation. This is important to note since it has been shown that the presence of a MixColumns operation in the last round would affect the security of AES (Dunkelman and Keller, 2010).

2.2 The Key Schedule

The key schedule generates a series of subkeys from the secret key. There are three variants of the AES corresponding to the three possible bit lengths of the secret key used, i.e. 128, 192 or 256 bits. In Algorithm 2 we show how the subkey bytes are generated from an initial secret key K. The function S is the substitution function used in the SubBytes operation described above. The function f is, for the most part, the identity function. However, when K is a 256-bit key and $j = 4$ then f is the substitution function S. RCON is a round constant that changes for each loop. We refer the reader to the AES specification for a more detailed description of the key schedule (FIPS PUB 197, 2001).

Algorithm 2: The AES key schedule function.

```
Input: $X$-bit secret key $K$, with $X$ set to 128,
       192 or 256 and $N$ set to 10, 12, 14
       respectively, RCON.
Output: $W$ a stream of subkey bytes.
for $i \leftarrow 0$ to $X/8-1$ do  $W[i] \leftarrow K[i]$ ;
for $i \leftarrow 1$ to $\lceil (N+1) \cdot (X/128)^2 \rceil$ do
    for $j \leftarrow 0$ to 3 do
        $W[i \cdot X + j] \leftarrow S(W[i \cdot (X-1) + j])$ ;
    end
    $W[i \cdot X] \leftarrow W[i \cdot X] \oplus$ RCON ;
    for $j \leftarrow 1$ to 3 do
        for $k \leftarrow 0$ to $X/4$ do
            $W[i \cdot X + 4\,j + k] \leftarrow W[i\,(X-1) + 4\,j$
            $+k] \oplus f(W[i \cdot X + 4\,(j-1) + k])$;
        end
    end
end
return $W$
```

For AES-128, knowing one subkey will allow the original key to be derived. For AES-192 and AES-256 there will still be some ambiguity and two subkeys are required to derive the original key.

3 THE SQUARE ATTACK

If we consider two plaintexts that have a XOR difference that is non-zero in one byte, then this difference will expand in a known manner. After one round the XOR difference between the intermediate states would show that one column of the state matrix has a non-zero difference. This property will then propagate to all the bytes in the state matrix after the second round by the same reasoning. An example where the difference in two plaintexts is at index one is shown in Figure 1.

It is therefore impossible that the XOR difference between two such plaintexts will be zero in any byte after two rounds. This property will persist until the next `MixColumns` operation, and is used for impossible differential cryptanalysis since the XOR difference after two rounds cannot be zero (Biham and Keller, 1999).

This property is also used to construct an attack referred to as the Square attack (so-called since it was first presented in the description of the block cipher Square (Daemen et al., 1997)) and was first presented in the original description of AES (Daemen and Rijmen, 1998). We consider 256 distinct plaintexts that are equal in fifteen bytes. After computing two rounds of AES the property described above will be valid between all possible pairs, i.e. across all 256 intermediate states the bytes at each index will contain one of each possible value. The XOR sum of the 256 bytes at each index will therefore be equal to zero. There will not be one instance of each possible value across the 256 bytes at each index after the next `MixColumns` operation. However, the XOR sum of the bytes at each index will still be zero after the `MixColumns` operation, and this property will remain true until the next `SubBytes` operation. In the remainder of this paper we will refer to a set of 256 chosen plaintext-ciphertext pairs where the 256 distinct plaintexts that are equal in fifteen bytes as a δ-set.

4 APPLYING THE SQUARE ATTACK TO AES-128

Attacks based on the Square attack applicable to AES-128 are presented in this section.

4.1 Analyzing Four-round AES

An attack based on the property described in Section 3 was originally detailed by Daemen and Rijmen in their AES proposal (Daemen and Rijmen, 1998). If we consider one δ-set, the XOR sum of the intermediate states at the end of the third round is equal to zero. For a four-round variant of AES an attacker can use this observation to validate hypotheses on the last subkey byte-by-byte, where an attacker checks that the XOR sum of the input to the final round is equal to zero. For each byte this will return the correct subkey byte and one additional incorrect hypothesis per byte with a probability of $255/256$. That is, any random sequence of byte will have an XOR sum equal to zero with a probability of $1/256$ and there are 255 such sequences. This will result in an expected total number of key hypotheses for the last subkey of $\left(1 + \frac{255}{256}\right)^{16} \approx 2^{16}$, since the length of the lists of hypotheses are mutually independent. One can determine the key if one repeats the analysis, i.e. one takes 2^9 chosen plaintexts and conducts the above analysis twice. This would have a time complexity of 2^9 one-round decryptions (2^7 encryptions of a four-round AES).

Biham and Keller observed the sum of a sequence of random bytes can be computed by only considering one example of values that occur with an odd-numbered frequency. Since values that occur with

$$\overset{\text{Plaintext}}{\begin{pmatrix}\zeta & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{pmatrix}} \rightarrow \overset{\text{First Round}}{\begin{pmatrix}2\theta & 0 & 0 & 0\\ 3\theta & 0 & 0 & 0\\ \theta & 0 & 0 & 0\\ \theta & 0 & 0 & 0\end{pmatrix}} \rightarrow \overset{\text{Second Round}}{\begin{pmatrix}2\alpha & \beta & \gamma & 3\delta\\ 3\alpha & 2\beta & \gamma & \delta\\ \alpha & 3\beta & 2\gamma & \delta\\ \alpha & \beta & 3\gamma & 2\delta\end{pmatrix}}$$

Figure 1: Propagation of a one-byte difference across two rounds of AES. A structure in this difference is imposed by the `MixColumns` operation (see Section 2.1).

an even-numbered frequency will have no effect on the XOR sum across all 256 intermediate values (Biham and Keller, 1999). Given 256 bytes taken from the same index from a δ-set, one can remove all values that occur with an even-numbered frequency and keep one example of those that occur with an odd-numbered frequency.

We define Z as the number of instances of a given value that occur in a sequence of 256 bytes. The probability of observing a given value n times is

$$\Pr(Z=n)=\binom{256}{n}256^{-n}\left(1-\frac{1}{256}\right)^{256-n} \qquad (1)$$

for $1\leq n\leq 256$. The probability of observing an odd number of a given value is therefore $\Pr(X=1)+\Pr(X=3)+\ldots+\Pr(X=255)=0.43$, and therefore the number of values that need to be treated decreases to $256\times 0.43=110$. The analysis given by Biham and Keller stops here and we provide a more precise analysis below.

We define Y as the number of distinct values that occur in a sequence of 256 bytes. The probability of observing m distinct values is

$$\Pr(Y=m)=\frac{256_{(m)}\left\{ {256 \atop m} \right\}}{256^{256}}\,, \qquad (2)$$

for $1\leq m\leq 256$. We define $r_{(m)}=r\,(r-1)\ldots(r-m+1)$ and $\left\{ {n \atop i} \right\}$ as a function that returns the Stirling numbers of the second kind. That is, the number of ways of partitioning n elements into i non-empty sets. The expectation of x is simply $\sum_{i=1}^{256} i\Pr(Y=i)=162$.

For a sequence of 256 bytes that consist of m distinct values, the probability distribution will be somewhat similar to that defined by Biham and Keller. Again we define Z as the number of instances of a given value that occur in a sequence of 256 bytes. The probability of observing observing a given value n times given that there are m distinct values is

$$\Pr(Z=n\,|\,Y=m)=\binom{256}{n}m^{-n}\left(1-\frac{1}{m}\right)^{256-n} \qquad (3)$$

for $1\leq m,n\leq 256$. Again, the probability of observing an odd number of a given value is therefore $\Pr(X=1\,|\,Y=m)+\Pr(X=3\,|\,Y=m)+\ldots+\Pr(X=255\,|\,Y=m)$ for a given m. We define A as the number of distinct values that occur with an odd-numbered frequency. Then the expectation of A will be:

$$E(A)=\sum_{i=1}^{256} i\Pr(Y=i)\sum_{j=0}^{127}\Pr(Z=(2\,j+1)\,|\,Y=i)\approx 78 \qquad (4)$$

That is, the sum of the number of distinct values occurring with an odd-numbered frequency $\left(i\sum_{j=0}^{127}\Pr(X=(2\,j+1)\,|\,Y=i)\right)$ multiplied by the probability of it occurring. This would reduce the time complexity of an attack requiring two δ-sets to 156 one-round decryptions (approximately 2^5 encryptions of a four-round AES).

4.2 Analyzing Five-round AES

An extension to the above attack was also first presented in the original description of AES (Daemen and Rijmen, 1998). This attack allowed an extra round to be analyzed with an increase in the time complexity. Rather than analyzing the final subkey byte-by-byte, one analyzes the penultimate subkey.

In order to do this one is obliged to guess 32 bits of the final subkey to determine one column of the state matrix before the XOR with the penultimate subkey. One can then compute the `MixColumns` operation on this column, and validate hypotheses on a byte of a subkey equivalent to the penultimate subkey (one could compute the `MixColumns` operation on the derived subkey to determine the penultimate subkey). A valid byte would allow the property described in Section 3 to be observed. Each evaluation reduces the potential key space of five bytes being analyzed by a factor of 256, and one would need to conduct this analysis five times to determine 32 bits of the final subkey (Daemen and Rijmen, 1998).

If we define each of the 2^{32} partial decryptions as having a time complexity equivalent to a quarter of a round, analyzing five sets of 256 ciphertexts to determine 32 bits of the last subkey and eight bits of the “penultimate” subkey, can be computed with an effort equivalent to $2^{40}/4$ one-round AES decryptions for one δ-set. Given this is a quarter of the work required for one set of 256 acquisitions, the total complexity to determine a key using five δ-sets would be

$5 \cdot 2^{40}$ one-round AES decryptions, or equivalent to 2^{40} five-round AES encryption operations.

The cryptanalysis cannot be significantly improved by following the reasoning given in Section 4.1. That is, if one partially decrypts a δ-set using hypotheses on 32 bits of the last subkey, one can then form hypotheses on individual bytes of the penultimate subkey using one example of distinct values that occur with an odd-numbered frequency. One could follow this reasoning with the 32-bit values taken from the ciphertexts. However, over 256 acquisitions the probability of observing a value that occurs with an even-numbered frequency will be too low to have any significant impact on the time complexity of an attack.

Ferguson et al. present a way of conducting this attack that is referred to as the “partial sums” method (Ferguson et al., 2001). They observe that conducting the attack involves computing

$$\sum_i S^{-1}(S_0\left(c_{i,0} \oplus k_0\right) \oplus S_1\left(c_{i,1} \oplus k_1\right) \oplus S_2\left(c_{i,2} \oplus k_2\right) \oplus S_3\left(c_{i,3} \oplus k_3\right) \oplus k_4), \tag{5}$$

where S_λ, for $\lambda \in \{0, \dots, 3\}$, are bijective look-up tables that consist of the function S and a multiplication by a field element from $\mathcal{F}_{2^8}$. These are evaluated efficiently by associating a “partial sum” x_k to each ciphertext where x_k is defined as

$$x_k \leftarrow \sum_{j=0}^{k} S_j\left(c_j \oplus k_j\right) . \tag{6}$$

This gives a map from $(c_0, c_1, c_2, c_3) \mapsto (x_k, x_{k+1}, \dots, c_3)$. In order to conduct an attack one can compute (x_1, c_2, c_3), i.e. $((S_0(c_0 \oplus k_0) \oplus S_1(c_1 \oplus k_1)), c_2, c_3)$, for all the ciphertexts in a δ-set for all possible values of k_0 and k_1. This will take 2^{24} executions of the function S resulting in 2^{24} values for (x_1, c_2, c_3). This continues by computing (x_2, c_3) for all possible values of k_2 that also requires a 2^{24} executions of the function S resulting in 2^{32} values for (x_2, c_3). Computing the 2^{40} values for x_3 for all values of k_3 will require a further 2^{40} executions of the function S. The last step will require 2^{48} executions of the function S^{-1}. Using the estimate provided by Ferguson et al., that the time complexity of one AES encryption is equivalent to 2^8 executions of the function S (Ferguson et al., 2001), implementing the attack described above will have a time complexity equivalent to approximately 2^{40} five-round AES encryption operations. This has the same time complexity as the straightforward approach described previously. The method reported by Ferguson et al. can only be applied when groups of δ-sets are treated together (see Section 4.3).

The above analyses assume that one only considers one byte of the subkey that is equivalent to the penultimate subkey. However, if we consider more bytes of the penultimate subkey then fewer δ-sets are required. An attacker guesses 32 bits of the last subkey which allows an attacker to validate hypotheses on four bytes of a subkey that is equivalent to the penultimate subkey. For one guess of 32 bits of the final subkey one would expect a hypothesis for any byte of the subkey that is equivalent to the penultimate subkey to produce an XOR sum equal to zero with a probability equal to $1/2^8$. Given that there are four such bytes the probability that four bytes of this subkey will produce four sequences with XOR sums equal to zero is $1/2^{32}$. Therefore, in order to determine 32 bits of the last subkey and 32 bits of a subkey equivalent to the penultimate subkey one would expect to need two δ-sets. This would reduce the time complexity of the attack detailed by Daemen and Rijmen (Daemen and Rijmen, 1998) to approximately 2^{39} five-round AES encryption operations.

The analysis of the hypotheses can be further optimized if the relationship between the last and penultimate subkey are verified as the attack progresses. If we consider one δ-set, one can analyze two columns of the penultimate subkey by guessing eight bytes of the last subkey (in two sets of four bytes). This will produce two sets of 2^{32} hypotheses with a time complexity equivalent to 2^{36} five-round AES encryption operations (using the estimations given above). One can then eliminate hypotheses in each set that are inconsistent, given that we have hypotheses on eight bytes of the penultimate key and eight bytes of the last subkey. However, we note that extra computation is required to change the hypotheses on the derived values to hypotheses for the penultimate subkey. That is, the four bytes corresponding to the penultimate subkey are multiplied by M as described in Section 2.1. This can be efficiently computed by considering the input vector as a 32-bit word and the matrix multiplication conducted using 32-bit operations. We estimate the complexity of this to be approximately equivalent to (5), which is equivalent to $1/2^6$ five-round AES encryption operations. Operating on these 2^{32} values will require the equivalent of 2^{26} five-round AES encryption operations, which is negligible compared to the generation of the hypotheses.

From the AES key schedule we can see that any subkey byte can be computed from two specific bytes in either the previous or following subkey. From the two sets of hypotheses generated, as described above, there will be three cases where known relationships between these sets of hypotheses can be verified. Given that the probability that all three bytes of a given hypotheses can be verified will be $1/2^{24}$, one would expect that the two sets of 2^{32} hypotheses can be reduced to one set of 2^{40} hypotheses. A third set of 2^{32} hypotheses can then be generated for one of the remaining columns of the penultimate subkey and four bytes of the last subkey. There will be a further four bytes of the last subkey that are generated by bytes for which there are already hypotheses, and an element from the set of of 2^{40} hypotheses will validate a hypothesis from the new set of 2^{32} hypotheses with a probability of $1/2^{32}$. One would therefore expect to combine these two sets to produce a set of 2^{40} hypotheses for 96 bits of the penultimate and 96 bits of the last subkey. A set of 2^{32} hypotheses can then be generated for the final column of the penultimate subkey and four bytes of the final subkey. At this point on can verify whether an entire subkey can be generated from the penultimate subkey. For each of the 2^{32} hypotheses generated, hypotheses in the set of 2^{40} hypotheses for 96 bits of the penultimate and 96 bits of the last subkey will produce valid keys with a probability of $1/(2^8)^9 = 1/2^{72}$ (since there will be nine bytes in the last subkey that will not have been verified previously). One would, therefore, expect to generate two hypotheses from the two sets of hypotheses. One that is correct and one that fulfills the criteria by chance.

The time complexity of the entire attack will be 2^{38} five-round AES encryption operations and require 2^{40} hypotheses to be stored in memory. If a second δ-set is included the time complexity will increase, but the memory requirements will become negligible. The time complexity does not double since an attacker only requires sufficient information to determine which of the two remaining hypotheses are false. This should be possible with work equivalent to 2^{36} five-round AES encryptions, i.e. the generation of hypotheses of 32-bits of the final and 32 bits of the penultimate subkeys. These attack are summarized in Table 1.

Table 1: Summary of the Square Attack on Five-round AES-128.

Number of δ-sets	Memory	Time Complexity	Remaining Hypotheses
1	2^{40}	2^{38}	2
2	1	2^{38}	1

4.3 Analyzing an Extra Round

The attack described above applied to a five-round AES can be extended to attack a six-round AES. In order to permit an extra round to be analyzed a set of 2^{32} plaintexts are chosen that give all the possible ciphertexts that differ at indexes $1, 6, 11$, and 16. An attacker will then seek to choose the 256 plaintext-ciphertext pairs that produce intermediate states that differ in only one byte after one round, i.e. the input required to attack a five-round AES, as shown in Figure 2.

The simplest way of achieving this would be to choose 256 32-bit values for the first column of the intermediate state that differ in one byte. These 32-bit values can be deciphered for a given hypothesis for four bytes of the first subkey (specifically the bytes at indexes $1, 6, 11$, and 16). This will produce 256 plaintexts that produce a δ-set after one round that can be analyzed using the attack described in Section 4.2, each of which will provide one hypothesis for the secret key given the hypotheses for 32 bits of the first subkey. This would increase the time complexity of an attack by a factor of 2^{32}, but allow an extra round to be analyzed.

Ferguson et al. observed that all 2^{32} can be used as a set of acquisitions to conduct the Square attack (Ferguson et al., 2001). That is, a set of 2^{32} distinct plaintexts differing at, for example, indexes $1, 6, 11$, and 16 described above can be viewed as 2^{24} δ-sets. This remains true after the first round but an attacker cannot distinguish individual δ-sets after the first round without knowing four bytes of the first subkey. However, an attacker can treat all 2^{32} acquisitions together, i.e. the attack described in the previous section work in the same manner but with a set of 2^{32}, rather than 2^8, acquisitions. We refer to a set of 2^{32} plaintext-ciphertext pairs that are equivalent to 2^{24} δ-sets as a Δ-set.

An attack would proceed in the same manner as described in Section 4.2. Using the same notation the computation of the sets of (x_1, c_2, c_3) will require 2^{48} executions of the function S. This would result in 2^{32} triples (x_1, c_2, c_3). However, a maximum of 2^{24} values distinct values are possible. As described in Section 4.1, one only needs to keep one example of the triplets that occur with an odd-numbered frequency. Likewise, this will produce at most 2^{16} values for (x_2, c_3) per key hypothesis, and at most 2^8 values for x_3 per key hypothesis. The time complexity of the entire analysis requires 2^{50} executions of the function S for all four 32-bit sets for the final subkey, which given our estimate given above corresponds to 2^{42} AES encryptions operations. This is increased to 2^{44} where five Δ-sets are required to determine the key.

Given that only two Δ-sets are required to determine the key, see Section 4.2, one the complexity using the "partial sums" method can be reduced to 2^{43}. This is not immediately apparent since evalu-

$$\overset{\text{Plaintext}}{\begin{pmatrix}\bullet & 0 & 0 & 0\\ 0 & \bullet & 0 & 0\\ 0 & 0 & \bullet & 0\\ 0 & 0 & 0 & \bullet\end{pmatrix}} \rightarrow \overset{\text{AddRoundKey}}{\begin{pmatrix}\bullet & 0 & 0 & 0\\ 0 & \bullet & 0 & 0\\ 0 & 0 & \bullet & 0\\ 0 & 0 & 0 & \bullet\end{pmatrix}} \rightarrow \overset{\text{SubBytes}}{\begin{pmatrix}\bullet & 0 & 0 & 0\\ 0 & \bullet & 0 & 0\\ 0 & 0 & \bullet & 0\\ 0 & 0 & 0 & \bullet\end{pmatrix}} \rightarrow \overset{\text{ShiftRow}}{\begin{pmatrix}\bullet & 0 & 0 & 0\\ \bullet & 0 & 0 & 0\\ \bullet & 0 & 0 & 0\\ \bullet & 0 & 0 & 0\end{pmatrix}} \rightarrow \overset{\text{MixColumn}}{\begin{pmatrix}\bullet & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{pmatrix}}$$

Figure 2: Propagation of a four-byte difference across one round of AES, where $\bullet$ represents a non-zero difference.

ating (5) will reduce the number of key hypotheses for $\{k_0, k_1, k_2, k_3, k_4\}$ by a factor of 256. Intuitively, one would expect that the same effort would be required to analyze one Δ-set four times, resulting in an attack with the same time complexity as that proposed by Ferguson et al. We note that evaluating (5) once for a given Δ-set will requires 2^{48} executions of the function S. A second evaluation of an instance of (5) with k_4 replaced with a different key byte from the penultimate subkey will require 2^{40} executions of the function S. This is because $S_\lambda(c_{i,\lambda} \oplus k_\lambda)$, for one $\lambda \in \{0,\ldots,3\}$, will already have been computed for all i (we note that this will involve carefully choosing the order in which the first instance of (5) is evaluated). This same reasoning applies for a third and fourth evaluation of (5) that will result in an 8-tuple consisting of four bytes of the last subkey and four bytes of the penultimate subkey. The overall complexity of evaluating (5) four times for one Δ-set is, therefore, approximately 2^{48} executions of the function S. The time complexity of the repeating this for all four 32-bit sets for the final subkey requires 2^{50} executions of the function S, which given our estimate given above corresponds to 2^{42} AES encryptions operations. This is increased to 2^{43} where two Δ-sets are required to determine the key.

We can reduce the time complexity further if we exploit the relationships between key bytes in adjacent subkeys. That is, one can generate 8-tuples of key byte values using the "partial sums" sums technique proposed by Ferguson et al. and use the attack described in in Section 4.2 to reduce the number of key hypotheses. Given one set of 2^{32} plaintext-ciphertext pairs an attack with a time complexity equivalent to 2^{42} six-round AES encryption operations and require that 2^{40} key hypotheses are stored in memory. Again, more acquisitions can be used to reduce the memory requirements at the cost of an increased time complexity. This is summarized in Table 2.

Table 2: Summary of the Square Attack on Five-round AES-128.

Number of δ-sets	Memory	Time Complexity	Remaining Hypotheses
1	2^{40}	2^{42}	2
2	1	2^{42}	1

5 APPLYING THE SQUARE ATTACK TO AES-192 AND AES-256

In this section we describe how the attacks described in Section 4 are applicable to AES-192 and AES-256.

5.1 Analyzing Five-round AES

Attacking a five-round AES-192 will function using a time complexity equivalent to 2^{39} AES encryption operations using the attack described in Section 4.2 based on the attack proposed by Ferguson et al. (Ferguson et al., 2001). That is, two δ-sets would be sufficient to determine the last two subkeys of a five-round instance of AES-192 or AES-256.

An attack on an instance of AES-256 cannot be made more efficient by analyzing two consecutive subkeys since there are no direct relationships. However, one can exploit the relationships between two consecutive subkeys when attacking an AES-192. This would follow a similar technique to that presented in Section 4.2. If one acquires a δ-set, one can analyze two columns of the penultimate subkey by guessing eight bytes of the last subkey (in two sets of four bytes). This will produce two sets of 2^{32} hypotheses with a time complexity equivalent to 2^{36} five-round AES encryption operations. One can then eliminate hypotheses in each set that are inconsistent, since we would have hypotheses on eight bytes of the penultimate key and eight bytes of the last subkey. Given the key schedule, one would expect that the two sets of 2^{32} hypotheses can be reduced to one set of 2^{48} hypotheses. A third set of 2^{32} hypotheses can then be generated for one of the remaining columns of the penultimate subkey and four bytes of the last subkey. One would expect that this would produce

2^{64} hypotheses with the relevant subkeys. This is because, in each case, there are two key bytes in the penultimate key that can each be derived from bytes the last subkey.

As previously, the memory requirements of the above attack can be reduced by acquiring more δ-sets at the cost of an increase in the time complexity. However, one would only need to analyse half the information present in a second δ-set to reduce the hypotheses to a trivial amount. The details of the square attack applied to five-round AES-192 and AES-256 are shown in Table 3 and 4 respectively.

Table 3: Summary of the Square Attack on Five-round AES-192.

Number of δ-sets	Memory	Time Complexity	Remaining Hypotheses
1	2^{64}	2^{38}	2^{64}
2	1	$2^{38.5}$	2^{16}

Table 4: Summary of the Square Attack on Five-round AES-256.

Number of δ-sets	Memory	Time Complexity	Remaining Hypotheses
1	2^{128}	2^{38}	2^{128}
2	1	2^{39}	1

5.2 Analyzing Six-round AES

The attacks on five rounds of AES-192 and AES-256 can be extended by assuming that an attacker knows the last subkey. The time complexity of the five-round attack does increases by a factor of 2^{128} when it is applied to a six round variants of AES-256. However, Lucks observed that knowledge of the last subkey would allow an attacker to directly derive bytes in previous subkeys when attacking AES-192 (Lucks, 2000). That is, 64 bits of the penultimate subkey and 32 bits of the antepenultimate subkey, as shown in Figure 3.

Following the reasoning in Section 4.2, a computational effort equivalent to 2^{22} (i.e. a factor of 2^{16} less than the standard attack) six-round AES-192 encryption operations to treat each δ-set. As previously, two δ-sets would be sufficient to determine the penultimate and antepenultimate subkeys.

One can reduce this to one set if one uses the relationships between two consecutive subkey as described in Section 5.1. Since much of the subkeys are already known the memory requirements are much reduced. The details of the square attack applied to five-round AES-192 and AES-256 are shown in Table 5 and 6 respectively.

Table 5: Summary of the Square Attack on Six-round AES-192 using δ-sets..

Number of δ-sets	Memory	Time Complexity	Remaining Hypotheses
1	2^{16}	2^{150}	2^{8}
2	1	2^{151}	1

Table 6: Summary of the Square Attack on Six-round AES-256 using δ-sets..

Number of δ-sets	Memory	Time Complexity	Remaining Hypotheses
1	2^{128}	2^{166}	2^{128}
2	1	2^{167}	1

We can also note that introducing an extra round before, and acquiring Δ-sets will lead to a more efficient attack. This involves conducting the five-round attack described in Section 5.1 on a six-round AES using Δ-sets rather than δ-sets. The details of the square attack applied to six-round AES-192 and AES-256 using Δ-sets are shown in Table 7 and 8 respectively.

Table 7: Summary of the Square Attack on Six-round AES-192 using Δ-sets.

Number of δ-sets	Memory	Time Complexity	Remaining Hypotheses
1	2^{64}	2^{42}	2^{64}
2	1	$2^{42.5}$	2^{16}

Table 8: Summary of the Square Attack on Five-round AES-256 using Δ-sets.

Number of δ-sets	Memory	Time Complexity	Remaining Hypotheses
1	2^{128}	2^{42}	2^{128}
2	1	2^{43}	1

Table 9: Summary of the Square Attack on Seven-round AES-192.

Number of δ-sets	Memory	Time Complexity	Remaining Hypotheses
1	2^{16}	2^{153}	2^{8}
2	1	2^{154}	1

Table 10: Summary of the Square Attack on Seven-round AES-256.

Number of δ-sets	Memory	Time Complexity	Remaining Hypotheses
1	2^{128}	2^{170}	2^{128}
2	1	2^{171}	1

$$\overset{\text{7-th Subkey}}{\begin{pmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{pmatrix}} \rightarrow \overset{\text{6-th Subkey}}{\begin{pmatrix} \circ & \circ & 0 & 0 \\ \circ & \circ & 0 & 0 \\ \circ & \circ & 0 & 0 \\ \circ & \circ & 0 & 0 \end{pmatrix}} \rightarrow \overset{\text{5-th Subkey}}{\begin{pmatrix} 0 & 0 & 0 & \circ \\ 0 & 0 & 0 & \circ \\ 0 & 0 & 0 & \circ \\ 0 & 0 & 0 & \circ \end{pmatrix}}$$

Figure 3: The known information from the last subkey, where $\bullet$ represents a known value and $\circ$ represents a derived value.

Table 11: Summary of attacks presented in this paper.

	Rounds	Key Length	Memory	Acquisitions	Complexity
(Daemen and Rijmen, 1998)	4	128	–	2^9	2^6
(Biham and Keller, 1999) (corrected)	4	128	–	2^9	2^5
(Daemen and Rijmen, 1998)	5	generic	–	2^{11}	2^{40}
This paper	5	128	–	2^8	2^{38}
This paper	5	192	–	$2 \cdot 2^8$	$2^{38.5}$
This paper	5	256	–	$2 \cdot 2^8$	2^{39}
(Daemen and Rijmen, 1998)	6	generic	–	$5 \cdot 2^{32}$	2^{72}
(Ferguson et al., 2001)	6	generic	2^{32}	$6 \cdot 2^{32}$	2^{44}
This paper	6	128	2^{40}	2^{32}	2^{42}
This paper	6	192	–	$2 \cdot 2^{32}$	$2^{42.5}$
This paper	6	256	–	$2 \cdot 2^{32}$	2^{43}
(Lucks, 2000)	7	192	2^{32}	2^{32}	2^{176}
Ferguson et al. (Ferguson et al., 2001)	7	192	2^{32}	$19 \cdot 2^{32}$	2^{155}
This paper	7	192	–	$2 \cdot 2^{32}$	2^{154}
(Lucks, 2000)	7	256	2^{32}	2^{32}	2^{192}
(Ferguson et al., 2001)	7	256	2^{32}	$21 \cdot 2^{32}$	2^{172}
This paper	7	256	–	$2 \cdot 2^{32}$	2^{171}

5.3 Analyzing an Extra Round

As described in Section 4.3, an extra round can be added where an attacker uses Δ-sets (Ferguson et al., 2001). Given that a large amount of the penultimate subkey is known the "partial sums" method of conducting the Square attack can be further optimized. We recall that the attack requires that (5) is evaluated as described in Section 4.2. Note that the attack does not proceed exactly as the six-round attack given in Section 5.2 as the relationships between the bytes of the subkeys will be different.

If, for example, an attacker wishes to evaluate (5) and knows (k_0, k_1) these values can be evaluated before the unknown (k_2, k_3). Generating 2^{24} values for (x_1, c_2, c_3) will require 2^{33} executions of the function S. Then 2^{16} values for (x_2, c_3) per hypotheses for k_2 can be generated with 2^{32} executions of the function S, and continue as per the attack described in Section 4.3. The time complexity of the entire analysis will be 2^{34} evaluations of the function S. That is, 2^{36} executions of the function S for all four 32-bit sets for the final subkey, which corresponds to 2^{28} AES encryptions operations.

6 CONCLUSIONS

In this paper we have demonstrated that the "partial sums" method of conducting the Square attack can be improved by analyzing more information per δ-set (or Δ-set). This allows the time complexity of attacks to be halved. For AES-128 the time complexity can be halved again, and reduced by a smaller amount for AES-192, by eliminating key hypotheses on-the-fly.

In Table 11 we present a summary of the attacks presented in this paper alongside similar attacks in the literature. The memory requirements are the number of state matrices, or equivalent, that need to be stored in memory. The number of acquisitions are the number of chosen plaintexts enciphered with an unknown key, where the number is given in multiples of 2^8 or 2^{32} so that it is clear how many δ or Δ-sets are required to conduct the attack. The time complexity is expressed as the computational effort required to compute that many AES encryption operations.

In this paper we have focused on attacks that require relatively few chosen plaintext-ciphertext pairs. Ferguson et al. also proposed a Square attack applicable to seven rounds of a AES-128 and eight rounds for AES-192 and AES-256. However, this attack requires approximately 2^{128} chosen plaintext-ciphertext pairs and is beyond the scope of this paper although one would expect the proposed strategy to aid in the analysis.

ACKNOWLEDGEMENTS

The author would like to thank Martijn Stam for helpful discussions. The work described in this paper has been supported in part the European Commission through the ICT Programme under Contract ICT-2007-216676 ECRYPT II and the EPSRC via grant EP/I005226/1.

REFERENCES

Bahrak, B. and Aref, M. R. (2008). Impossible differential attack on seven-round AES-128. *IET Information Security Journal*, 2(2):28–32.

Biham, E. and Keller, N. (1999). Cryptanalysis of reduced variants of Rijndael. unpublished. http://www.madchat.fr/crypto/codebreakers/35-ebiham.pdf.

Bogdanov, A., Khovratovich, D., and Rechberger, C. (2011). Biclique cryptanalysis of the full AES. In Lee, D. H. and Wang, X., editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 344–371. Springer.

Daemen, J., Knudsen, L., and Rijmen, V. (1997). The block cipher Square. In Biham, E., editor, *FSE '97*, volume 1267 of *LNCS*, pages 149–165. Springer.

Daemen, J. and Rijmen, V. (1998). AES proposal: Rijndael. In *AES Round 1 Technical Evaluation CD-1: Documentation*. NIST. http://www.nist.gov/aes.

Dunkelman, O. and Keller, N. (2010). The effects of the omission of last round's MixColumns on AES. *Information Processing Letters*, 110(8–9):304–308.

Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., and Whiting, D. (2001). Improved cryptanalysis of Rijndael. In Schneier, B., editor, *FSE 2000*, volume 1978 of *LNCS*, pages 213–230. Springer.

FIPS PUB 197 (2001). Advanced encryption standard (AES). Federal Information Processing Standards Publication 197, National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA.

Lu, J., Dunkelman, O., Keller, N., and Kim, J. (2008). New impossible differential attacks on AES. In Chowdhury, D. R., Rijmen, V., and Das, A., editors, *INDOCRYPT 2008*, volume 5365 of *LNCS*, pages 279–293. Springer.

Lucks, S. (2000). Attacking seven rounds of Rijndael under 196-bit and 256-bit keys. In *AES Candidate Conference 2000*. http://csrc.nist.gov/encryption/aes/round2/conf3/aes3conf.htm.

Mangard, S., Oswald, E., and Popp, T. (2007). *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer-Verlag.

Zhang, W., Wu, W., and Feng, D. (2007). New results on impossible differential cryptanalysis of reduced AES. In Nam, K.-H. and Rhee, G., editors, *ICISC 2007*, volume 4817 of *LNCS*, pages 239–250. Springer.