Key Schedule Classification of the AES Candidates

G. Carter‡ E. Dawson† L. Nielsen†

† Information Security Research Centre
‡ School of Mathematical Sciences
Queensland University of Technology
GPO Box 2434 Brisbane 4001
Queensland Australia

Abstract

An important component of iterative, block ciphers is the key schedule. In most ciphers, a master key of specified length is manipulated to create round subkeys. This manipulation is known as the *key schedule*. A strong key schedule means a cipher will be more resistant to various forms of attacks, such as differential and linear cryptanalysis. In this paper, the Advanced Encryption Standard(AES) candidates are classified according to their key schedules.

1 The Classification Schedule

The most powerful methods of analysis of iterative block ciphers such as the Data Encryption Standard(DES) [4] are attacks which aim to reveal round subkeys. These methods include differential [5] and linear cryptanalysis [9].

In [1], the authors introduced a new classification scheme for iterative block ciphers based on their key schedules. In essence, this scheme creates two categories of ciphers based on whether or not knowledge of a round subkey generated by the key schedule reveals any information about other round subkeys or the master key. Those that do, fall into *Category 1* and those that do not, fall into *Category 2*. Each of these categories is further subdivided into three Types, A, B and C.

A Category 1, Type A cipher(1A) is one in which all bits of the master key are used in each round, and hence knowledge of a round subkey yields all bits of the master key and all other round subkeys. The cipher NDS [3] is such an example. At the other end of the scale, 2C ciphers are those in which each round subkey is generated independently, and the length of the master key is the sum of the lengths of all the round subkeys. The cipher DESI(DES with independent subkeys) [5] is an example.

A 1B cipher is one where knowledge of a round subkey gives some, but not all bits of the master key or other round subkeys. DES is an example. A 1C cipher is one in which knowledge of a round subkey yields bits of other round subkeys or the master key after some simple arithmetic operations or function inversions. SAFER K-64 [6] is an example.

In Category 2, knowledge of a round subkey does not easily reveal information about other round subkeys or the master key. A 2A cipher is one in which not all bits of the master key are used to create each round subkey. In these ciphers, certain master keys are guaranteed to produce at least two identical round keys. A cipher such as CAST-128 [7] is an example. In other words, the entropy of the round subkeys is not maximised. A 2B cipher is one in which all master key bits are used in the determination of all round subkeys, thus maximising the entropy of the subkeys. An example is Blowfish [8].

The most secure schedule classification is 2C. However, this may lead to unmanageably large master keys for ciphers whose security cannot hope to match what is naively suggested by the key length. Further, export restrictions on cryptographic materials often limit the size of the key. For these reasons, the best we can hope for is to mimic 2C schedules as closely as possible, with the next strongest classification, 2B.

2 Classification of AES Candidates

In this section, all the AES candidates are classified according to the key schedule classification described in the previous section, and the justification for their placement in a certain category presented. Although many of these candidates permit parameter values outside the scope of the AES standard, the analysis presented below will assume a 128-bit block size and a 128-bit, 192-bit or 256-bit master key. The analysis of each of the AES candidates will assume the 256-bit master key option and comments will be made on the other two master key options only where the master key length has ramifications for the classification.

For each of the AES ciphers, an outline of the key schedule will be presented and the following two questions posed.

1. *Given knowlege of all the bits of a round subkey, does this reveal any bits of other round subkeys or the master key?*
2. *Do all round subkeys depend on all bits of the master key?*

The answer to the first question will place the cipher in either Category 1 or 2. The answer to the second question will determine whether the Category 2 ciphers are Type A or B. Note that no AES candidate is Category 2, Type C. The results of the classification appear in Table 1. The justification for the classification of each cipher according to this scheme is presented below. The description of the AES candidates was obtained from [2].

CAST-256

The 256-bit master key can be described as eight, 32-bit *words*, $b_0^0 b_1^0 \cdots b_7^0$. By setting low order bytes to zero, other master key lengths can be obtained. For example, if b_6^0 and b_7^0 are set to zero, the master key has length one hundred and ninety-two bits.

To determine the j'th round subkey each b_i^{j-1} is modified as follows:

$$b_i^j = b_i^{j-1} \oplus f(b_{i+1 (\mathrm{mod}\ 8)}, c_j, d_j),\ 0 \le i \le 7,\ 1 \le j \le 12$$

where f is one of three functions used in the cipher, and c_j and d_j are deterministic constants. The round subkey is, in fact, a key pair (k_r^j, k_m^j). Key k_r^j is the

Type 1A	*Type 1B*	*Type 1C*	*Type 2A*	*Type 2B*	*Type 2C*
	MAGENTA	**CRYPTON** **DEAL** **Rijndael** **SAFER+**	**DFC**	**CAST-256** **E2** **FROG** **HPC** **LOKI97** **MARS** **RC6** **Serpent** **Twofish**	

Table 1: Classification of Key Schedules

concatenation of five least significant bits(LSB) of each of the modified words b_0^j, b_2^j, b_4^j and b_6^j while k_m^j is the concatenation of modified words b_7^j, b_5^j, b_3^j and b_1^j. Knowing the round subkey key pair (k_r^j, k_m^j) gives knowledge of words b_7^j, b_5^j, b_3^j and b_1^j as well as the five LSB of each of b_0^j, b_2^j, b_4^j and b_6^j. This is not enough information to determine the previous round subkey. Hence **CAST-256** is a Type 2 cipher. However, it is worth pointing out, that if the remaining unknown bits of b_0^j, b_2^j, b_4^j and b_6^j can be determined, then all previous round subkeys and the master key can be found.

To generate the subkey pair, (k_r^j, k_m^j), all words b_0^j to b_7^j are used. As these words are dependent on all the original master key words, subkey pairs are dependent on all master key words. This leads to the conclusion that **CAST-256** is a 2B cipher.

CRYPTON

The master key, if not two hundred and fifty-six bits, is prepended by zeros to make it so. Using a mixture of linear and non-linear transformations, eight, 32-bit expanded keys, $E_0, E_1 \cdots, E_7$ are produced from the original master key. All subkeys are two hundred and fifty-six bits, the first and second subkeys respectively being, $K_0 = (K^3, K^2, K^1, K^0) = (E_0, E_1, E_2, E_3)$ and $K_1 = (K^7, K^6, K^5, K^4) = (E_4, E_5, E_6, E_7)$. Subsequent even round subkeys are given by, $K_{2i+2} = (K^{8i+11}, K^{8i+10}, K^{8i+9}, K^{8i+8}),\ 0 \le i \le 5$ and odd round subkeys are given by $K_{2j+3} = (K^{8j+15}, K^{8j+14}, K^{8j+13}, K^{8j+12}),\ 0 \le j \le 4$. Now,

$$
\begin{aligned}
K^{8i+8} &= K^{8i} \lll (8i),\ i \in \{0,2,4\} \\
K^{8i+8} &= RC_i \oplus K^{8i},\ i \in \{1,3,5\} \\
K^{8i+9} &= RC_i \oplus K^{8i+1},\ i \in \{0,2,4\} \\
K^{8i+9} &= K^{8i+1} \lll (3i^2 - 20i + 41),\ i \in \{1,3,5\} \\
K^{8i+10} &= K^{8i+2} \lll (-3i^2 + 10i + 16),\ i \in \{0,2,4\} \\
K^{8i+10} &= RC_i \oplus K^{8i+2},\ i \in \{1,3,5\} \\
K^{8i+11} &= RC_i \oplus K^{8i+3},\ i \in \{0,2,4\} \\
K^{8i+11} &= K^{8i+3} \lll (4i+4),\ i \in \{1,3,5\} \\
K^{8j+12} &= RC_j \oplus K^{8j+4},\ j \in \{0,2,4\}
\end{aligned}
$$

$$
\begin{aligned}
K^{8j+12} &= K^{8j+4} \ll (4j+4),\ j \in \{1,3\} \\
K^{8j+13} &= K^{8j+5} \ll (-3j^2+10j+16),\ j \in \{0,2,4\} \\
K^{8j+13} &= RC_j \oplus K^{8j+5},\ j \in \{1,3\} \\
K^{8j+14} &= RC_j \oplus K^{8j+6},\ j \in \{0,2,4\} \\
K^{8j+14} &= K^{8j+6} \ll (4j+12),\ j \in \{1,3\} \\
K^{8j+15} &= K^{8j+7} \ll (3j^2-14j+24),\ j \in \{0,2,4\} \\
K^{8j+15} &= RC_j \oplus K^{8j+7},\ j \in \{1,3\}
\end{aligned}
$$

where RC_x is a known constant and $x \ll y$ is a known left shift of x by y bits.

Knowing even round subkey $(K^{8i+11}, K^{8i+10}, K^{8i+9}, K^{8i+8})$ means that the following can be shown true by *undoing* the above equations, as follows.

$$
\begin{aligned}
K^{8i} &= K^{8i+8} \gg (8i),\ i \in \{0,2,4\} \\
K^{8i} &= RC_i \oplus K^{8i+8},\ i \in \{1,3,5\} \\
K^{8i+1} &= RC_i \oplus K^{8i+9},\ i \in \{0,2,4\} \\
K^{8i+1} &= K^{8i+9} \gg (3i^2-20i+41),\ i \in \{1,3,5\} \\
K^{8i+2} &= K^{8i+10} \gg (-3i^2+10i+16),\ i \in \{0,2,4\} \\
K^{8i+2} &= RC_i \oplus K^{8i+10},\ i \in \{1,3,5\} \\
K^{8i+3} &= RC_i \oplus K^{8i+11},\ i \in \{0,2,4\} \\
K^{8i+3} &= K^{8i+11} \gg (4i+4),\ i \in \{1,3,5\}
\end{aligned}
$$

Thus, the previous even round subkey is easily determined. A similar argument applies to odd round subkeys. It follows that knowing one even/odd round subkey enables all previous even/odd round subkeys to be determined by simple arithmetic operations. It is also worth noting that by using exactly the same arguments, knowledge of an even/odd round subkey enables all subsequent even/odd round subkeys to be determined. Hence **CRYPTON** is a 1C cipher.

DEAL

The key schedule of **DEAL** is created from two, three or four, 64-bit DES keys, corresponding to 128-bit, 192-bit and 256-bit master keys. The 128-bit and 192-bit master keys are used in six rounds of **DEAL**, while the 256-bit version uses eight rounds. In all three cases, the creation of the round subkeys is essentially identical.

The eight round subkeys are generated from four independent master keys, designated K_1, K_2, K_3 and K_4, as follows.

$$
\begin{aligned}
RK_1 &= DESE_K(K_1) \\
RK_2 &= DESE_K(K_2 \oplus RK_1) \\
RK_3 &= DESE_K(K_3 \oplus RK_2) \\
RK_4 &= DESE_K(K_4 \oplus RK_3) \\
RK_5 &= DESE_K(K_1 \oplus \{1\} \oplus RK_4) \\
RK_6 &= DESE_K(K_2 \oplus \{2\} \oplus RK_5) \\
RK_7 &= DESE_K(K_3 \oplus \{4\} \oplus RK_6) \\
RK_8 &= DESE_K(K_4 \oplus \{8\} \oplus RK_7)
\end{aligned}
$$

Note that $\{i\}$ denotes the representation of i as a 64-bit string and $DESE_K(X)$ represents the encryption of X using DES with key K which is fixed and known.

Knowing one value of $RK_i,\ i \in \{2,3,4,5,6,7,8\}$ does not immediately yield any bits of other subkeys or the master keys despite the fact that the decryption $DESD_K(RK_i)$ can easily be performed since K is known. On the other hand, knowledge of RK_1 easily yields the master key K_1, since $K_1 = DESD_K(RK_1)$. Hence RK_1 is much weaker than the other subkeys and puts **DEAL** in the 1C classification.

Further, RK_1 depends only on master key K_1, RK_2 depends only on master keys K_1 and K_2, and RK_3 depends only on master keys K_1, K_2 and K_3. The other RK_i's depend on all four master keys. Hence, not all subkeys depend on all bits of the master key.

If the known key K is replaced by K_1, K_2, K_3 or K_4 then this would eliminate the weak RK_1 and the cipher would fall into the 2A category. Another alternative would be to let $K = f(K_1, K_2, K_3, K_4)$ where f is a one way function(OWF) and this would put **DEAL** into the 2B category.

Decorrelated Fast Cipher-DFC

The master key K is initially padded on the right with a 256-bit known constant, C_0. The result is then truncated to form padded master key K_C, so that what remains is two hundred and fifty-six bits long. This means that if K is one hundred and twenty-eight bits then only the one hundred and twenty-eight most significant bits(MSB) of C_0 are used to form K_C. Similarly, if K is one hundred and ninety-two bits, the sixty-four MSB of C_0 are used. Finally, if K is two hundred and fifty-six bits, no bits of C_0 are required.

At this point, K_C is divided into eight, 32-bit strings such that $K_C = K_C^1|K_C^2|\cdots|K_C^8$. Note that $X|Y$ means X concatenated with Y.
Now the following are defined.

$$\begin{aligned} OAP_1 &= K_C^1|K_C^8 \\ OBP_1 &= K_C^5|K_C^4 \\ EAP_1 &= K_C^2|K_C^7 \\ EBP_1 &= K_C^6|K_C^3 \end{aligned}$$

and for $i = 2, 3$ and 4

$$\begin{aligned} OAP_i &= OAP_1 \oplus C_1^i \\ OBP_i &= OBP_1 \oplus C_2^i \\ EAP_i &= EAP_1 \oplus C_1^i \\ EBP_i &= EBP_1 \oplus C_2^i \end{aligned}$$

where C_1^i and C_2^i are known constants.
Now define

$$EF_1(K) = (F_1|F_2|F_3|F_4)$$

where $F_i = OAP_i|OBP_i$. Similarly, define

$$EF_2(K) = (f_1|f_2|f_3|f_4)$$

where $f_i = EAP_i|EBP_i$. Now define $RK_0 = 0$ and create round subkeys $RK_i,\ 0 \le i \le 8$, as follows.

$$RK_i = \begin{cases} \text{DFC}_{EF_1(K)}(RK_{i-1}) & \text{if } i \text{ is odd} \\ \text{DFC}_{EF_2(K)}(RK_{i-1}) & \text{if } i \text{ is even,} \end{cases}$$

where $\text{DFC}_{EF_i(K)}(X)$ means encrypt X using key $EF_i(K)$ in the DFC cipher.

Now, knowledge of RK_i gives no bits of any other round subkeys unless the encryption process can be reversed. On the other hand, the generation of RK_i, i odd involves only $EF_1(K)$ which in turn involves F_i, $i \in \{1,2,3,4\}$. Now F_i involves only OAP_i and OBP_i. These in turn involve only K_C^1, K_C^4, K_C^5 and K_C^8, and not all bits of K_C or indeed K. A similar scenario applies for RK_i, i even. Thus, not all subkeys depend on on all bits of the master key. Hence, **DFC** is a Category 2, Type A cipher.

E2

Define a 64-bit constant C. If the length of the master key, K, is one hundred and twenty-eight bits, considered as the concatenation of K_1 and K_2, then pass C through a given set of S-boxes three times to produce the 64-bit value K_3 and then a fourth time to produce 64-bit K_4. In this process, the output of the S-boxes on each pass becomes the input for the next pass. If a 192-bit master key is used, considered as the concatenation of K_1, K_2 and K_3, repeat the above process to produce K_4. If a 256-bit master key is in use then C is not processed as above. Let the key resulting from the above procedure be $KA = K_1|K_2|K_3|K_4$. The round subkeys are then formed as follows.

Firstly, KA and C are put into a function G used in the cipher such that the output $(L_0, (Z_0, C_0)) = G((K_1, K_2, K_3, K_4), C)$ where $L_0 = (U_1, U_2, U_3, U_4)$, $Z_0 = (Y_1, Y_2, Y_3, Y_4)$, and $C_0 = V$, and for $i = 1, 2, 3, 4$

$$\begin{aligned} Y_i &= f(K_i) \\ U_i &= f(U_{i-1}) \oplus Y_i \text{ and } U_0 = C \\ V &= U_4 \end{aligned}$$

where f is a function used in the cipher. Note that U_i, Y_i and V are all 64-bit values.

For $i = 0, 1, \cdots, 7$ define

$$\begin{aligned} (L_{i+1}, (Z_{i+1}, C_{i+1})) &= G(Y_i, C_i) \\ L_{i+1} &= (l_{4i}, l_{4i+1}, l_{4i+2}, l_{4i+3}) \end{aligned} \tag{1}$$

Next i

where l_i is a 64-bit block.

For $i = 1, 2, \cdots, 31$ define

$l_i = (t_i^0, t_i^1, \cdots, t_i^7)$

Next i

Each 128-bit subkey is now generated as follows.

For $i = 0, 1, \cdots, 15$,

$$k_{i+1} = (t^{<i/2>}_{0+(i \bmod 2)}, t^{<i/2>}_{2+(i \bmod 2)}, \cdots, t^{<i/2>}_{30+(i \bmod 2)})$$

Next i

where $< x >$ means the greatest integer $\leq x$.

Knowledge of subkey k_i, i even, yields only eight bits out of a possible fifty-six from each of l_{2j+1}, $j = 0, 1, \cdots, 15$ and hence only sixteen bits out of a possible two hundred and fifty six bits of each L_i. In any L_i, only the bits involved in k_i are known, and these do not reveal any of the other bits of the L_i's involved with other round subkeys. A similar argument applies when k_i is known and i is odd. Further, there is no simple way to invert equation 1 which is required to track back to the master key. Thus, knowledge of a round subkey yields no bits of other round subkeys or the master key.

An examination of the way L_1 is constructed will show that it depends on all master key bits and hence all subsequent L_i's depend on L_1. Since round subkey i depends on L_i, it follows that all round subkeys depend on L_1 and hence on all bits of the master key.

Now $L_1 = G(Z_0, C_0)$, that is $G((Y_1, Y_2, Y_3, Y_4), U_4)$. As well, $U_4 = f(U_3) \oplus f(K_4)$, $U_3 = f(U_2) \oplus f(K_3)$, $U_2 = f(U_1) \oplus f(K_2)$ and $U_1 = f(C) \oplus f(K_1)$. Thus, U_4 depends on all bits of the master key. Now $L_1 = (U_1', U_2', U_3', U_4')$ where $U_1' = f(U_4) \oplus f(Y_1)$. Since U_4 depends on all bits of the master key so does U_1' and since U_i' depends on U_{i-1}', L_1 depends on all bits of the master key. All the L_i's contribute to the determination of subkey one, so subkey one depends on all bits of the master key.

Now L_i depends on Y_{i-1} which in turn depends on L_{i-1}, so L_i depends on L_{i-1} and hence on L_1 and all bits of the master key. The conclusion is that **E2** is a 2B cipher.

FROG

The master key, K, in this cipher is essentially hashed to produce a 2304-byte valid internal key, KIV. KIV can be thought of as the concatenation eight, 288-byte round subkeys. Each of these subkeys has three records, and each of these records is used in a different way to encrypt the plaintext. The hashing process begins by concatenating copies of the master key until a 2304-byte array has been produced. The elements of this array are then XOR'd with a randomly generated, but known constant. This *unformatted* array is then *formatted* so that the resulting array is a preliminary version of the KIV. This formatting is necessary so that the three records mentioned earlier achieve rapid diffusion and confusion. The formatted array is then encrypted by the **FROG** algorithm itself to further the random appearance of the subkeys. This encrypted output is again formatted to satisfy the reqiurements of the subkey records.

Knowledge of a round subkey does not reveal any bits of other round subkeys, since the 288-byte blocks of KIV which determine each subkey are generated at the same time and are essentially independent. Thus the **FROG** cipher belongs in Category 2, Type B.

Hasty Pudding Cipher-HPC

Assuming a 256-bit master key, K, divide it into four, 64-bit words, K_0, K_1, K_2 and K_3. Now construct a 256-element array, KX, each element, a 64-bit word. The first three elements, $KX[0], KX[1]$ and $KX[2]$ are respectively initialised to known constants c_0, c_1 and c_2. The remaining elements of the array are constructed as follows.

For $i = 3, 4, \cdots, 255$

$$KX[i] = KX[i-1] + (KX[i-2] \oplus KX[i-3] \gg 23 \oplus KX[i-3] \ll 41)$$

Next i,

where $x \gg y$ and $x \ll y$ are respectively quantity x right shifted/left shifted by

y bits. Master key K is now introduced into KX as follows.

For $i = 0, 1, \cdots, 127$

$$KX[i] = KX[i] \oplus K_{i \bmod 4}$$

Next i

The resulting elements are then *stirred* as follows.

State variables, $s_j, j = 0, 1, \cdots, 7$ are initialised respectively to $KX[248 + j]$ and the following series of steps are performed in each of three passes over the array KX.

For each $KX[i], i = 0, 1, \cdots, 255$

$$\begin{aligned}
s_0 &= (KX[i] \oplus KX[(i+1) \wedge 255]) + KX[s_0 \wedge 255] \\
s_1 &= s_1 + s_0 \\
s_3 &= s_3 \oplus s_2 \\
s_5 &= s_5 - s_4 \\
s_7 &= s_7 \oplus s_6 \\
s_3 &= s_3 + (s_0 \ SL \ 13) \\
s_4 &= s_4 \oplus (s_1 \ SL \ 11) \\
s_5 &= s_5 \oplus (s_3 \ SL \ (s_1 \wedge 31)) \\
s_6 &= s_6 + (s_2 \ SR \ 17) \\
s_7 &= s_7 \vee (s_3 + s_4) \\
s_2 &= s_2 - s_5 \\
s_0 &= s_0 - (s_6 \oplus i) \\
s_1 &= s_1 + c_0 \\
s_2 &= s_2 + (s_7 \ SR \ j) \\
s_2 &= s_2 \oplus s_1 \\
s_4 &= s_4 - s_3 \\
s_6 &= s_6 \oplus s_5 \\
s_0 &= s_0 + s_7 \\
KX[i] &= s_2 + s_6
\end{aligned}$$

Note that the operations $+$ and $-$ are performed modulo 2^{64}, $\wedge$ means *logical and*, $\vee$ is *logical or*, SR and SL are right and left shifts respectively, and $j = 0, 1, 2$ represents the pass number over the array KX.

Suppose $KX[i]$ is known. From the last line in the set of steps above, $s_2 + s_6$ is known. However, there is no obvious way to determine either s_2 or s_6. Even if these were known, the nature of the stirring process means that it is not invertible, so no bits of the master key or other round subkeys can be obtained from a knowledge of $KX[i]$.

Now $s_i, i = 0, 1, \cdots, 7$ contains two copies of the master key arranged such that any four s_i values, say s_k, s_l, s_m and s_n, with $k \not\equiv l \not\equiv m \not\equiv n (\bmod 4)$, contain all bits of the master key. From the last line of the above series of steps, $KX[i]$ depends on s_2 and s_6. Tracing back through this series of steps it is easy to see that s_6 depends on s_5, s_3, s_1 and s_0 and that s_5 is related to s_3, s_1 and s_4. Hence, s_6 depends on at least s_0, s_1 and s_3. Thus, $KX[i]$ depends at least on s_0, s_1, s_2 and s_3, and since $0 \not\equiv 1 \not\equiv 2 \not\equiv 3 (\text{mod } 4), KX[i]$ depends on all bits of the master key. Thus, all round subkeys depend on all bits of the master key. This puts the **HPC** cipher in the 2B class.

Because the 'spice' is not necessarily a secret key, it has been deliberately omitted from this discussion.

LOKI97

The master key, K is initialised into four, 64-bit words, $K_1^0|K_2^0|K_3^0|K_4^0$, as follows.

256-bit $K = K_1|K_2|K_3|K_4$ yields $K_1^0|K_2^0|K_3^0|K_4^0 = K_1|K_2|K_3|K_4$.

192-bit $K = K_1|K_2|K_3$ yields $K_1^0|K_2^0|K_3^0|K_4^0 = K_1|K_2|K_3|f(K_1, K_2)$ where f is a non-linear function used in the encryption process.

128-bit $K = K_1|K_2$ yields $K_1^0|K_2^0|K_3^0|K_4^0 = K_1|K_2|f(K_2, K_1)|f(K_1, K_2)$.
These initialised keys are then processed as follows to yield forty-eight, 64-bit round subkeys, SK_i.

For $i = 1, \cdots, 48$

$$\begin{aligned} K_1^i &= K_4^{i-1} \oplus g_i(K_1^{i-1}, K_3^{i-1}, K_2^{i-1}) \qquad (2) \\ SK_i &= K_1^i \\ K_4^i &= K_3^{i-1} \\ K_3^i &= K_2^{i-1} \\ K_2^i &= K_1^{i-1} \end{aligned}$$

Next i

Note that $g_i(K_1, K_3, K_2) = f(K_1 + K_3 + \delta i, K_2)$ where δ is a constant and f is a function used in the cipher.

Knowledge of SK_i does not reveal any bits of previous or subsequent round subkeys, as these are dependent on at least other three other unknown subkeys or initialised master keys. Further, the generation of subkey SK_1 involves all bits of the master key as per equation 2. Since the generation of subkey SK_i depends on subkey SK_{i-1}, SK_i ultimately depends on SK_1 and hence on all bits of the master key. Thus, **LOKI97** is a 2B cipher.

MAGENTA

For a master key, K, of one hundred and twenty-eight bits, this cipher has six rounds under the control of the master key considered as two, 64-bit blocks, K_1 and K_2. In rounds one, two, five and six of the encryption process, K_1 is used. In rounds four and five, K_2 is used.

In the 192-bit version of $K = K_1|K_2|K_3$, K_i is used in rounds i and $7 - i$.

In the 256-bit version, $K = K_1|K_2|K_3|K_4$, and the cipher has eight rounds with K_i being used in rounds i and $9 - i$. Clearly, knowledge of a round subkey yields immediately bits of the master key, as in DES. Thus, **MAGENTA** is a 1B cipher.

MARS

In this cipher, forty, 32-bit subkeys are required. An array, A, of forty-seven, 32-bit words is initialised using an n-word master key, k, as follows.

For $i = -7, -6, \cdots, -1$,
$A[i] = S[i+7]$
Next i

where $S[j]$ is the j'th entry of a fixed and known S-box.

For $i = 1, 2, \cdots, 38$,
$A[i] = ((A[i-7] \oplus A[i-2]) \lll 3) \oplus k[i \bmod n] \oplus i$
Next i
$A[39] = n$

This initialised array is further *stirred* as follows.

For $j = 1, 2, \cdots, 7$
For $i = 1, 2, \cdots, 39$
$A^j[i] = (A^{j-1}[i] + S_9[A^{j-1}[i-1]]) \lll 9$
Next i
$A^j[0] = (A^{j-1}[0] + S_9[A^j[39]]) \lll 9$
Next j

Note that $S_9[x]$ is the S-box entry indexed by the nine LSB of x. The subkeys are then created as follows.

For $i = 0, 1, 2, \cdots, 39$
$A[i]$ becomes subkey $K[7i \bmod 40]$
Next i

Consider now the case of $n = 8$, that is a 256-bit master key. In the initialization of array A, $A[i]$ depends on array words $A[i-7]$ and $A[i-2]$ as well as master key word $k[i \bmod 8]$. A careful analysis shows that not until the creation of $A[13]$ are the elements of array A dependent on all eight words of the master key. Subsequent elements, with the exception of $A[39]$, are also dependent on all master key words. $A[39]$ does not depend on any bits of the master key at all.

Now suppose an attacker knows $K[i] = A^{(7)}[j] = (A^{(6)}[j] \oplus S_9^[A^{(6)}[j-1]]) \lll 9$, for some j. There exists keys $K[h]$ and $K[g]$ such that $K[h] = A^{(7)}[j-1] = (A^{(6)}[j-1] \oplus S_9^{(6)}[A[j-2]]) \lll 9$ for some j, and $K[g] = A^{(7)}[j+1] = (A^{(6)}[j+1] \oplus S_9^[A^{(6)}[j]]) \lll 9$ for some j. Knowledge of $K[i]$ does not reveal $A^{(6)}[j]$ or $S_9[A^{(6)}[j-1]]$ which are components of $K[h]$ and $K[g]$. Neither does it reveal any information on $A^{(7)}[j-1]$ or $A^{(6)}[j+1]$, the other components of $K[h]$ and $K[g]$ respectively. Similarly, knowledge of $K[i]$ does not reveal any information about the master key.

The 7-round stirring process esentially combines successive entries in array A to create a new array $A^{(7)}$. After the first round of stirring, only elements $T^{(1)}[1]$ through to $T^{(1)}[6]$ inclusive do not depend on all bits of the master key. In each of the remaining six rounds of stirring, $T^{(2)}[1], T^{(3)}[2], T^{(4)}[3], T^{(5)}[4], T^{(6)}[5]$ and $T^{(7)}[6]$ successively come to depend on all master key bits. Hence, all subkeys depend on all bits of the master key. Thus, **MARS** is a 2B cipher.

RC6

This cipher has a precursor named RC5 [**?**]. The key schedule of **RC6** is identical to that of RC5 with the exception that a total of $2r+4$ subkeys are required for **RC6** while RC5 requires only $2r+2$ subkeys(r is the number of rounds of the cipher). The proposed number of rounds of **RC6** is twenty, so the number of subkeys required is forty-four. The subkeys are generated as follows.

From a master key of sixteen, twenty-four or thirty-two bytes, an array L of four, six or eight, 32-bit words is constructed. A second array S of forty-four, 32-bit words is initialised by known constants. The following loop is then executed one hundred and thirty-two times(3×44), after the variables i, j, A and B are initialised to zero.

For $s = 1$ to 132
    $S[i] = (S[i] + A + B) \lll 3$
    $A = S[i]$
    $L[j] = (L[j] + A + B) \lll (A + B)$
    $B = L[j]$
    $i \equiv (i + 1) (\bmod 44)$
    $j \equiv (j + 1) (\bmod c)$
Next s

Note that $c = 4, 6$ or 8 when the master key used is 128-bit, 192-bit or 256-bit respectively.

Suppose now that round key $S^3[i] = (S^2[i] + A + B) \lll 3$ is known. From this knowledge, it is not possible to determine any of the quantities that are involved in the calculation of $S^3[i]$. Even if these quantities were known, it would not be possible to determine, from a knowledge of these, $S^3[i+1]$ or $S^3[i-1]$, as the quantities which form these are related to those of $S^3[i]$ in a very *difficult to invert* way. Hence, knowledge of a round subkey does not yield any bits of any other round subkey or the master key.

Further, each entry in the array S is updated three times in the above loop. At the end of the first update, call it S^1, corresponding to $s = 44, S^1[0]$ to $S^1[7]$ inclusive are the only elements of array S^1 that are not dependent on all elements of the array L, that is, all elements of the master key. By the end of the second pass, all elements of S^2 are dependent on all elements of the master key. It follows that **RC6** is a 2B cipher.

Rijndael

This cipher stores its master key, K, in a $4 \times i$ array of bytes, where $i = 4, 6$ or 8 when K is a 128-bit, 192-bit or 256-bit key respectively. Given that the AES standard supports a 128-bit block size, the number of subkeys required is $\frac{128}{4} \times (r + 1)$, where r is the number of rounds. For a 128-bit master key, $r = 10$, for a 192-bit master key, $r = 12$ and for a 256-bit master key, $r = 14$.

An expanded key, K_e, is formed from the 256-bit master key array, $A_{x,y}, 0 \leq x \leq 3,\ 0 \leq y \leq 7$ as follows. Note that elements of array A are bytes.

An array, W, with $4 \times (14 + 1) (= 60)$ elements is constructed by setting $W[t],\ t = 0, 1, \cdots, 7$ to be successively equal to the four bytes of the master key, $A_{0,t}, A_{1,t}, A_{2,t}, A_{3,t}$. Subsequent elements of array W are constructed by the recursive relationship:

For $j = 8, j < 60$ and j incremented by 8
    $W[j] = W[j-8] \oplus f(W[j-1]) \oplus g(c_j)$
        For $l = 1, 2$ and 3
            $W[l + j] = W[l + j - 8] \oplus W[l + j - 1]$
        Next l
    $W[j + 4] = W[j - 4] \oplus h(W[j + 3])$
        For $l = 5, 6$ and 7

$$W[l+j] = W[l+j-8] \oplus W[l+j-1]$$
Next l

The functions f, g and h are functions associated with encryption and decryption and c_j is a deterministic constant.

For $x = 0, 1, 2, \cdots, 14$, round subkey x is given array entries $W[8x]$ through to $W[8x+7]$.

Now suppose that round subkey KS_x is known. It follows that $W[8x], W[8x+1], \cdots, W[8x+7]$ are known. Subkey KS_{x+1} consists of array elements $W[8x+8], W[8x+9], \cdots, W[8x+15]$. Now

$$W[8x+8] = W[8x] \oplus f(W[8x+7] \oplus g(c_x)) \tag{3}$$

All of the quantities on the right hand side of equation 3 are known since they are from the known subkey x or are constants. Hence $W[8x+8]$ is known. Since $W[8x+p], p = 9, 10, \cdots, 15$ depends in a simple way on $W[8x+p-1]$ and $W[8x+p-8]$ which is known from round subkey RK_x, calculating $W[8x+p]$ is easy. Hence, knowing subkey RK_x enables subkey RK_{x+1} to be easily determined. In fact, knowing subkey RK_x enables all subsequent subkeys to be determined. The conclusion is that the 256-bit master key version of the cipher **Rijndael** is 1C.

SAFER+

For 128-bit, 192-bit and 256-bit master keys K, the respective number of rounds in the cipher is eight, twelve and sixteen. The number of subkeys required is respectively seventeen, twenty-five and thirty-three. An analysis of the case when a 256-bit master key will be presented.

Firstly, thirty-three, 16-byte *bias* words, B_i, are created as follows.

For $i = 0, 1, \cdots, 15$

$$B_i^j \equiv 45^{(45^{17i+j+18} \bmod 257)} \pmod{257} \tag{4}$$

Next i,

where B_i^j is the j'th byte of bias word B_i and $j = 0, 1, \cdots, 15$. If the value of B_i^j, calculated in equation 4, is 256, then B_i^j is set to zero. The remaining bias words B_{17} to B_{32} inclusive are calculated as follows.

For $i = 17, 18, \cdots, 32$

$B_i^j \equiv 45^{17i+j+18} \pmod{257}$

Next i

Note that B_0 is a *dummy* and is not used at all.

A 33-byte word, K_e, is initialised by concatenating the thirty-two bytes of the master key K and the byte formed by the XOR sum of the corresponding bits of the thirty-two bytes of the master key. This last byte is known as the *parity* byte. The bytes of K_e will be denoted by $b_0, b_1, \cdots, b_{31}, b_{32}$. Round i uses two, 16-byte subkeys, K_i and K_{i+1}. The round one subkey, K_0, is calculated as follows.

$$K_0 = b_0 | b_1 | \cdots | b_{15}.$$

The concatenation of x_i through to x_j will be written as $C_{k=i}^{j}(x_k)$. Hence $K_0 = C_{k=0}^{15}(b_k)$. Subsequent subkeys are generated as follows.

For $i = 1, 2, \cdots, 32$
  For $j = 0, 1, \cdots, 32$
    $b_j = b_j \lll 3$
  Next j
  $K_i = C_{k=i}^{(i+15) \bmod 33}(b_k + B_i^{(k-i) \bmod 33})(\bmod 256)$
Next i.

Now suppose round subkey, K_i, is known. Hence,

$$C_{k=i}^{(i+15) \bmod 33}(b_k + B_i^{(k-i) \bmod 33})(\bmod 256)$$

is known. Since all the biases B_i are predetermined constants, b_k is simply calculated as $(K_i^{(k-i) \bmod 33} - B_i^{(k-i) \bmod 33})(\bmod 256))$. Thus, bytes b_i to $b_{(i+15) \bmod 33}$ are known. Right shifting each of these bytes by three bits, yields fifteen of the sixteen bytes used in in the calculation of K_{i-1}. Left shifting these known bytes by three bits will yield fifteen of the sixteen bytes used in the calculation of K_{i+1}. Thus, fifteen bytes of subkeys K_{i-1} and K_{i+1} are easily determined. In fact, if the known bytes of K_i are right shifted by an amount $3i$, then either fifteen or sixteen bytes of the master key will be known. Sixteen will be known if b_{33} is not one of the sixteen known bytes of K_i. On the other hand, if b_{33} is one of the known bytes of K_i, then fifteen bytes of the master key can be determined. Hence **SAFER+** with a 256-bit master key is a 1C cipher.

Serpent

This cipher requires thirty-three, 128-bit, round subkeys, and these are created from one hundred and thirty-two, 32-bit, interim words which are generated from the master key, K. If the master key is not two hundred and fifty-six bits in length, it is padded until it is. This padded master key, K_p, can be thought of as eight, 32-bit words labelled $W_{-8}, W_{-7}, \cdots, W_{-1}$. The one hundred and thirty-two, interim words are calculated as follows.

For $i = 0, 1, \cdots, 131$

$$W_i = (W_{i-8} \oplus W_{i-5} \oplus W_{i-3} \oplus W_{i-1} \oplus \phi \oplus i) \lll 11 \quad (5)$$

Next i,

where ϕ is a known constant.

Now suppose that subkey, K_i, is known.

Hence $X = (W_{4i}, W_{4i+1}, W_{4i+2}, W_{4i+3})$ is known. Each component of X is essentially the XOR sum of three previously calculated W_j values. The removal of the constants ϕ and i from the calculation is trivial(see equation 5). Thus, each component on its own does not reveal any information about any other W_j's and hence does not reveal any information about other round subkeys. Further, the XOR linear combination of any number of the components of X, does not isolate any single W_j value used in another round subkey. Hence, knowledge of a round subkey does not reveal any bits of other round subkeys or the master key.

Now, round i subkey, K_i, is the concatenation of $k_{4i}, k_{4i+1}, k_{4i+2}, k_{4i+3}$, where

$$(k_{4i}, k_{4i+1}, k_{4i+2}, k_{4i+3}) = S_{(3-i) \bmod 7}(W_{4i}, W_{4i+1}, W_{4i+2}, W_{4i+3}) \quad (6)$$

and S_x is an S-box of the cipher. From equation 6, the round i subkey, K_i, depends on $W_{4i}, W_{4i+1}, W_{4i+2}, W_{4i+3}$. From equation 5 the following three statements are true.
1. W_0 depends on key words W_{-8}, W_{-5}, W_{-3} and W_{-1}.
2. W_1 depends on key words W_{-7}, W_{-4}, W_{-2} and W_0 and since the dependencies of W_0 are known from 1, W_1 depends on $W_{-8}, W_{-7}, W_{-5}, W_{-4}, W_{-3}, W_{-2}$ and W_{-1}.
3. W_2 depends on key words W_{-6}, W_{-3}, W_{-1} and W_1 and since the dependencies of W_1 are known from 2, W_2 depends on $W_{-8}, W_{-7}, W_{-6}, W_{-5}, W_{-4}, W_{-3}, W_{-2}$ and W_{-1}, that is all bits of the master key.

Since from equation 5, W_i depends on key word W_{i-1}; for $i > 2, W_i$ will ultimely depend on W_2 and hence on all bits of the master key. Since every subkey K_i is dependent on a $W_i, i > 2$, all subkeys are dependent on all bits of the master key. This analysis classes **Serpent** as 2B.

Twofish

Twofish employs sixteen rounds with two, 32-bit subkeys in each round. Eight other subkeys are required. Four are added to the plaintext before encryption and the other four are added to the output of the last round of encryption. Thus, forty round subkeys are required. As in previous ciphers, the case of a 256-bit master key will be examined in detail.

Master key K is split into eight, 32-bit words, $K_0, K_1, \cdots, K_7$ where $K_i = m_{4i+3}|m_{4i+2}|m_{4i+1}|m_{4i}$ and m_j is the j'th byte of K. Let $M_e = K_0|K_2|K_4|K_6$, the concatenation of the even index K_j's, and $M_o = K_1|K_3|K_5|K_7$, the concatenation of the odd index K_j's. Define constants A_i and B_i as follows.

$$\begin{aligned} A_i &= h(2i\rho, M_e) \\ B_i &= ROL(h((2i+1)\rho, M_o), 8) \end{aligned}$$

Note that ρ is a defined constant, h is a function used in the cipher and $ROL(x, y)$ means rotate left, the quantity x, by y bits.

The forty, round subkeys, K^i, are created as follows.

For $i = 0, 1, \cdots, 19$

$$K^{2i} = (A_i + B_i)(\text{mod } 2^{32}) \tag{7}$$

$$K^{2i+1} = ROL((A_i + 2B_i)(\text{mod } 2^{32}), 9) \tag{8}$$

Next i

Suppose K^j is known. From equations 7 and 8, there is no obvious way to determine A_i and B_i and hence M_e and M_o, which are required, if other round subkeys are to be found from a knowledge of K^j. Thus, knowledge of a round subkey does not enable bits of other round subkeys or the master key to be determined.

Clearly, each round subkey is a function of A_i and B_i which respectively are functions of M_e and M_o. M_e and M_o encompass all bits of the master key, so it follows likewise for A_i and B_i. Hence, all round subkeys depend on all bits of the master key. It is now possible to classify **Twofish** as a 2B cipher.

However, it is worth noting that if two successive subkeys, K^{2i} and K^{2i+1} are known, subtracting(mod 2^{32}) K^{2i} from $ROR(K^{2i+1}, 9)$ will give B_i. From this, A_i can be determined from equation 8. With A_i and B_i now known, it is possible to invert the function h and hence determine M_e and M_o which comprise the master key.

3 Conclusion and Future Research

This paper is in response to calls by NIST for the analysis of the strengths and weaknesses of the AES candidates. It concentrates solely on the key schedules and makes no comment on the strengths and weaknesses of the algorithms or other aspects of the ciphers. The majority of candidates fall into the 2B, key schedule classification and these exhibit stronger key schedules than those that do not. For those in the other classifications, it is recommended that they be upgraded to the 2B category. In some cases, this is easily achieved as explained in the DEAL cipher. For the others, the natural extension of this paper is to upgrade them to the 2B class, and this is planned.

References

[1] Carter G., Dawson E., Nielsen L., *Key Schedules of Iterative Block Ciphers*, Information Security and Privacy ACISP'98, LCNS 1438, pp. 80-89, 1998.

[2] *Candidate Algorithms*, http://www.nist.gov/aes.

[3] Beker H., Piper F. *Cipher systems: the protection of communications*, Northwood Books, London, 1982.

[4] National Bureau of Standards, *Data Encryption Standard*, US Department of Commerce, FIPS publication 46, 1977.

[5] Biham E., Shamir A. *Differential Cryptanalysis of DES-like cryptosystems*, Advances in Cryptology-Crypto'90, LNCS 537, pp 2-21, Springer, 1991.

[6] Massey J., *SAFER K-64: a byte-oriented block ciphering algorithm*, Fast Software Encryption, LNCS 809, Springer-Verlag, 1994, pp 1-17.

[7] Adams C.M., *Constructing Symmetric Ciphers Using the CAST Design Procedure*, Designs, Codes and Cryptography, Vol. 12, No. 3, Nov. 1997.

[8] Schneier B., *Description of a new variable-length key, 64-bit block cipher(Blowfish)*, Fast Software Encryption, LNCS 809, Springer-Verlag, 1993, pp. 191-204.

[9] Matsui M., *Linear cryptanalysis method for DES cipher*, Advances in Cryptology, Eurocrypt'93, LNCS 765, Springer-Verlag, 1994, pp. 386-397.

[10] Rivest R., *The RC5 encryption algorithm*, Fast Software Encryption, LNCS 1008, Springer-Verlag, 1995, pp. 86-96.