Differential Fault Analysis on AES with 192 and 256-Bit Keys

Junko Takahashi and Toshinori Fukunaga
NTT Information Sharing Platform Laboratories
Nippon Telegraph and Telephone Corporation
3-9-1 Midori-cho, Musashino-shi, Tokyo, 180-8585, JAPAN
{takahashi.junko, fukunaga.toshinori}@lab.ntt.co.jp

Abstract

This paper describes a differential fault analysis (DFA) on AES with 192 and 256-bit keys. We show a new attack in which both 192 and 256-bit keys are retrieved within a feasible computational time. In order to verify the proposed attack and estimate the calculation time, we implement the proposed attack using C code on a PC. As a result, we successfully recover the original 192-bit key using 3 pairs of correct and faulty ciphertexts within 5 minutes, and 256-bit key using 2 pairs of correct and faulty ciphertexts and 2 pairs of correct and faulty plaintexts within 10 minutes. [1]

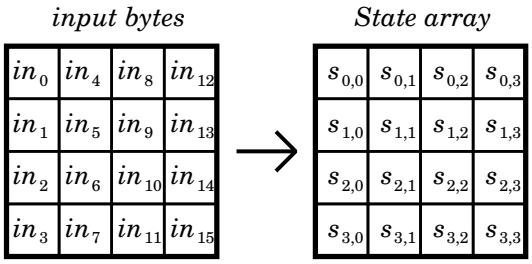

Figure 1. AES State.

1 Introduction

A side-channel attack is considered to be serious because the secret keys embedded in a secure computing device such as a smart card and RFID tag can be recovered within a feasible computational time. Fault analysis is one type of side-channel attack that deduces the secret key by deliberately inducing faults into the secure device during its cryptographic computation. Differential fault analysis (DFA) proposed by Biham *et al.* [1] is the most well known in fault analysis. In their attack, the secret key of DES can be recovered by comparing the correct and faulty output results after injecting faults into the secure device. Until now, DFA attacks against some symmetrical ciphers have been proposed and have had success in recovering secret keys [1, 2, 3].

There are many papers that propose a DFA attack against AES where the secret keys are recovered by inducing faults into the data part [4, 5, 6, 7, 8] or key expansion part [9, 10, 11]. They improve the attack conditions such as the fault model and requiring pairs of correct and faulty ciphertexts needed to recover the secret key. However, all previous attacks can only be applied to AES with a 128-bit key and the attack against 192 or 256-bit key has not been proposed in the published paper.

In this paper, we propose the attack against AES with 192 and 256-bit key. Our attack uses the characteristic structure of the key expansion routines for 192 and 256-bit keys and we can retrieve the original key within a practical time. In order to verify the proposed attack and evaluate the total calculation time needed to obtain the original key, we implement the proposed attack using C code on a PC. The simulation results show that we can retrieve the 192-bit key using 3 pairs of correct and faulty ciphertexts within 5 minutes, and we can retrieve the 256-bit key using 2 pairs of correct and faulty ciphertexts and 2 pairs of correct and faulty plaintexts within 10 minutes.

The reminder of this paper is organized as follows. Notations are defined in Section 2. In Section 3, we describe the previous work. We describe the idea of the proposed attack in Section 4. We describe the attack assumptions in Sections 5 and the attack procedures in Section 6 and 7. The simulation results are shown in Section 8. Finally, we conclude the paper in Section 9.

2 Notations

We define some of the parameters used in this paper.

The AES algorithm operations are performed on a two-

[1] This paper is the English version of the publication in the Japanese domestic symposium, Symposium on Cryptography and Information Security, SCIS 2010, which will be held on Jan. 19-22, 2010.

dimensional array of bytes called the State in Fig.1(See [12] for more details). At the start of the Cipher and Inverse Cipher, the input – the array of bytes $in_0, in_1, \ldots in_{15}$ – is copied into the State array as illustrated in Fig.1.The Cipher or Inverse Cipher operations are then conducted on the State array.

We use the parameters as follows.

$a_{(n)}$:	n denotes the bit length of a
$a \oplus b$:	Bitwise exclusive-OR operation
$a \cdot b$:	Finite field multiplication
K^n :	n-th round key that consists of 16 bytes, 4 by 4 matrix
$K^n_{i,j}$:	(i, j) byte of K^n $(i, j = 0 \ldots 3)$, where i and j are the row and column numbers

3 Previous Work

In this section, we briefly introduce well-known previous attacks against AES with a 128-bit key [6, 7] and we show the difficulty in applying these attacks to 192 and 256-bit keys.

The previous works use the fact that the state difference between the states calculated from correct and faulty ciphertexts is related to another state difference in the same column because one byte fault injected into the state is propagated to four bytes due to the MixCoumns transformation. We obtain the last round key K^{10} by solving the simultaneous equations related to the state differences by 4 bytes using correct and faulty ciphertexts [2]. Once we obtain K^{10}, we also obtain the original 128-bit key.

We verify we can extend the above attack method to the case of 192/256-bit key. In order to obtain the 192-bit key (256-bit key), we need to obtain the last round key K^r and the right half of K^{r-1} (whole K^{r-1}). We obtain K^r as the attack for 128-bit key. We can not obtain K^{r-1} by simply applying the above attack, because we have to deduce K^{r-1} by 16 bytes (not 4 bytes) due to the influence of the MixColumns transformation of the $r-1$ round. Therefore, the simultaneous equations related to K^{r-1} are more complicated, and we can not solve these equations in a practical time.

4 Idea of Proposed Attack

In this section, we describe the idea of the proposed attack.

[2] In fact, we obtain K^{10} by 4 bytes using three kinds of simultaneous equations.

In the case of 192-bit key We find that a part of the 10th round output difference between correct and faulty calculations are known because we obtain the left-half of K^{11} using K^{12} by the characteristics of the key expansion routine. Further, we find that the known output difference has a relation to the output difference calculated from the unknown right-half of K^{11}. Therefore, we can solve the simultaneous equations of the 10th round output difference by 4 bytes, and we can obtain the right-half of K^{11} within a practical time. Therefore, we obtain the original 192-bit key using the right-half of K^{11} and K^{12}.

In the case of 256-bit key In the case of 256-bit key, we can not obtain even 1 byte of K^{13} using K^{14} by its structure of the key expansion routine. Then, we have to solve the simultaneous equations by 16 bytes, and it is difficult to deduce the original key within a practical time. Here, we find that the first round key K^0 can be uniquely determined when we assume the value of one column (4 bytes) of K^{13}. by the characteristics of the key expansion routine. We verify whether the assumed one column (4 bytes) of K^{13} is correct or not by determining K^0 using the fault injections into the decryption. Using this method, we reduce the key candidate space of K^{13}. Therefore, we obtain the original 256-bit key using K^{13} and K^{14}.

In the next section, we propose the attack using the above idea.

5 Attack Assumptions

In this section, the attack assumptions are described.

- Correct and faulty ciphertexts calculated from the same plaintext are known.
- One pair of correct plaintext and ciphertext is known.
- Any one column (total 4 bytes) of the rth round output is randomly corrupted.
  - In the case of 192-bit key: $r = 9, 10$
  - In the case of 256-bit key: $r = 12$ (encryption and decryption)

  To satisfy the above conditions, fault injection area can be chosen as follows.
  - Any 1 byte of the $(r-1)$th round output.
  - Any one column (total 4 bytes) of the rth round output.
- (For the attack against 256-bit key) Correct and faulty plaintexts calculated from the same ciphertext are known.

6 Attack Procedure for 192-Bit Key

In this section, we describe the attack procedure for 192-bit key. In order to obtain the original 192-bit key, we need to deduce the right-half of K^{11}, $(K^{11}_{0,i}, K^{11}_{1,i}, K^{11}_{2,i}, K^{11}_{3,i})$ $(i = 2, 3)$ and K^{12}. Using the proposed attack, we reduce the key candidate space of these values to 2^8. Finally we obtain the original 192-bit key by the exhaustive search.

6.1 Deduce K^{12}

We obtain K^{12} using the method described in [13].

In order to obtain K^{12}, we obtain two kinds of faulty ciphertexts by injecting the faults into 1 byte of the 9th round output. The left figure of Fig.2 shows the fault propagation when 1 byte fault is injected into the 9th round output $s_{0,0}$. We can deduce K^{12} by solving the simultaneous equations of K^{12} within 1 minute using 2 pairs of correct and faulty ciphertexts on a PC.

6.2 Obtain faulty ciphertext

We obtain the faulty ciphertext by fault injection into 1 byte of the 8th round output (or one column (total 4 bytes) of the 9th round output). As an example, we explain the case when the fault is injected into 1 byte of the 8th round output $s_{0,0}$ in the following paper.

6.3 Deduce the left-half of K^{11}

As we know K^{12}, we can deduce 8 bytes of the left-half of K^{11}, $(K^{11}_{0,i}, K^{11}_{1,i}, K^{11}_{2,i}, K^{11}_{3,i})$ $(i = 0, 1)$ by the characteristics of the key expansion routine (Fig. 3).

6.4 Deduce the right-half of K^{11}

We deduce 8 bytes of the right-half of K^{11}, $(K^{11}_{0,i}, K^{11}_{1,i}, K^{11}_{2,i}, K^{11}_{3,i})$ $(i = 2, 3)$. Figure 4 shows the last part of the encryption for 192-bit key. The bytes shown in gray color are known values because we calculate them from K^{12} deduced in Section 6.1, the left-half of K^{11} deduced in Section 6.3, and the known ciphertext. Using these values, we deduce the 10th round output differences between correct and faulty encryption shown in gray color in Fig.4.

Moreover, the 10th round output differences between correct and faulty encryption shown in the box with the hatched line can be uniquely determined when we assume the 4 byte values of K^{11}, $(K^{11}_{0,3}, K^{11}_{1,3}, K^{11}_{2,3}, K^{11}_{3,3})$ shown in the box with the hatched line in Fig.4.

Here, due to the linearity of the MixColumns transformation, the input and output differences of the MixColumns transformation are held. Then, the differences of the byte

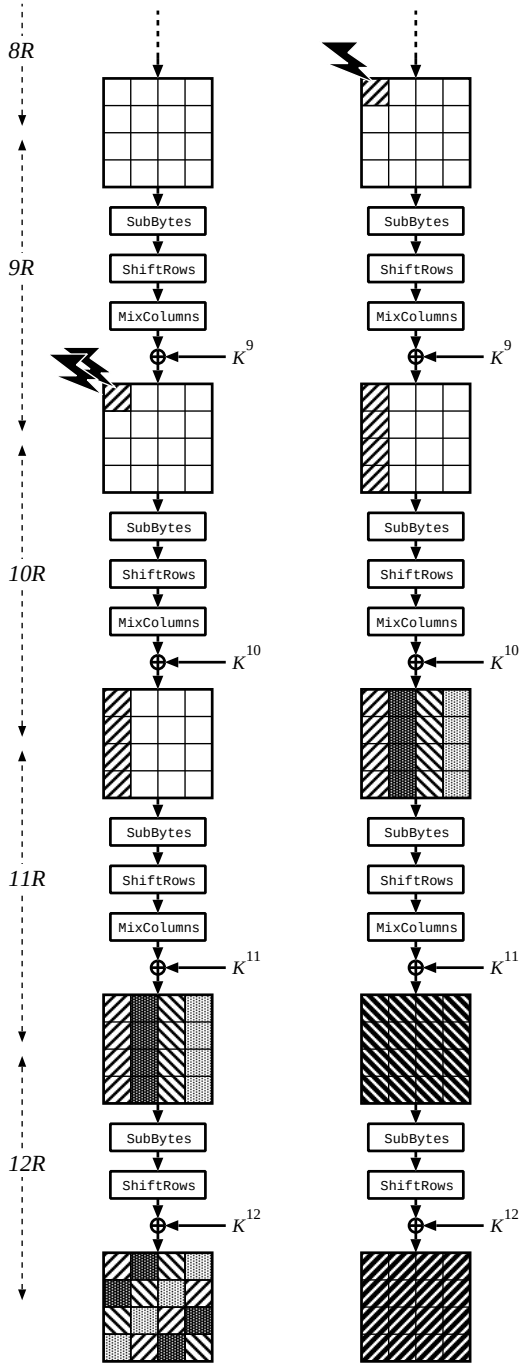

Figure 2. (Left) Fault propagation when the fault is injected into 1 byte of output in the 9th round. (Right) Fault propagation when the fault is injected into 1 byte of output in the 8th round.

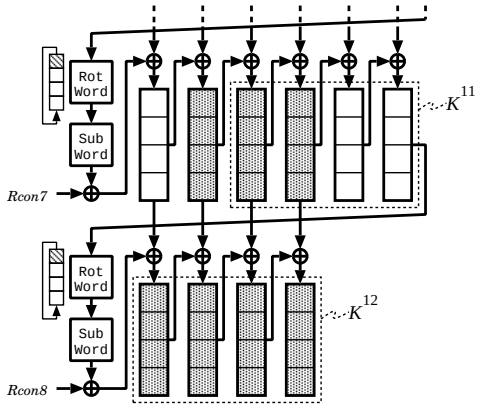

Figure 3. Last part of the key schedule for 192-bit key

between correct and faulty encryption have a relation to another byte in the same column, because the one byte fault is propagated to four bytes due to the influence of the MixColumns transformation. As an example, the most left column of the 10th round output satisfy the following equations.

$$M_{0,0} \oplus \widetilde{M}_{0,0} = \{02\} \cdot F \quad (1)$$

$$M_{1,0} \oplus \widetilde{M}_{1,0} = F \quad (2)$$

$$M_{3,0} \oplus \widetilde{M}_{3,0} = \{03\} \cdot F \quad (3)$$

In the above, F is the fault value. The notation $M_{i,j}$ ($\widetilde{M}_{i,j}$) is the correct (faulty) 10th round output. Further, the following simultaneous equations are satisfied.

$$M_{0,0} \oplus \widetilde{M}_{0,0} = \{02\} \cdot (M_{1,0} \oplus \widetilde{M}_{1,0}) \quad (4)$$

$$M_{3,0} \oplus \widetilde{M}_{3,0} = \{03\} \cdot (M_{1,0} \oplus \widetilde{M}_{1,0}) \quad (5)$$

Here, $M_{1,0}$ can be calculated from the above equations because the left parts of eq.(4) and (5) are known values. The byte $M_{1,0}$ is calculated from the guessed values of ($K^{11}_{0,3}$, $K^{11}_{1,3}$, $K^{11}_{2,3}$, $K^{11}_{3,3}$). Then, we can reduce the space of the key candidates of ($K^{11}_{0,3}$, $K^{11}_{1,3}$, $K^{11}_{2,3}$, $K^{11}_{3,3}$) by eliminating the guessed values which does not satisfy eq.(4) and (5). The above method can be applied from the second column to fourth column of 10th round output. Then, the state $M_{2,1}$, $M_{3,2}$, $M_{0,3}$ are also calculated from the above guessed values ($K^{11}_{0,3}$, $K^{11}_{1,3}$, $K^{11}_{2,3}$, $K^{11}_{3,3}$). We also reduce the space of the key candidates by eliminating the key candidates which does not satisfy the equations of $M_{2,1}$, $M_{3,2}$, $M_{0,3}$.

In the same way, we reduce the space of the key candidates ($K^{11}_{0,2}$, $K^{11}_{1,2}$, $K^{11}_{2,2}$, $K^{11}_{3,2}$). Therefore, we deduce the key candidates of the right-half of K^{11} using 1 pair of correct and faulty ciphertext.

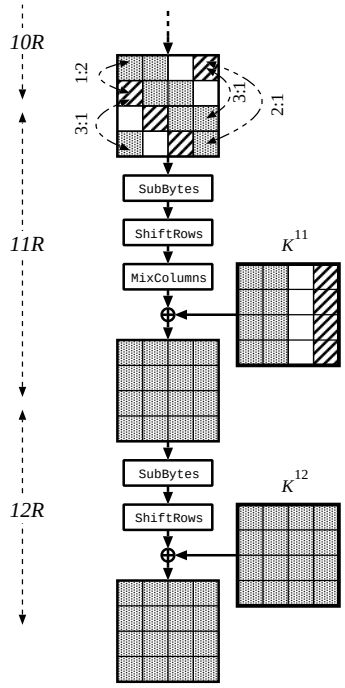

Figure 4. Data part for 192-bit key

6.5 Deduce the original 192-bit key

Using K_{12} deduced in Section 6.1 and the key candidates K^{11} deduced in Section 6.4, we calculate the key candidates of the original 192-bit key $k_{(192)}$ by applying the inverse calculation of the key expansion routine. If the candidates of the ciphertext calculated from $k_{(192)}$ is equal to the correct ciphertext, the key candidates of the original 192-bit key is correct and equal to the original key $K_{(192)}$.

7 Attack procedure for 256-bit key

In this section, we describe the attack procedure for 256-bit key. In order to obtain the original 256-bit key, we need to deduce K^{13} and K^{14}. Using our attack, we reduce the key candidate space of these values to 2^{13}. Finally, we obtain the original 256-bit key by the exhaustive search.

7.1 Deduce K^{14}

As Section 6.1, we deduce K^{14}. The last round key K^{14} is uniquely determined using 2 pairs of correct and faulty ciphertexts.

7.2 Deduce K^0

In order to obtain K^0, we obtain two kinds of faulty plaintexts by injecting the faults into 1 byte of the 11th round output. As Section 6.1, the first round key K^0 is uniquely determined using 2 pairs of correct and faulty plaintexts.

7.3 Deduce K^{13}

As we know K^{14}, we also know bytes indicated by the gray color in Fig.5 by calculating the inverse of the key expansion routine.

Here, we guess 4 bytes of $(K^{13}_{0,0}, K^{13}_{0,1}, K^{13}_{0,2}, K^{13}_{0,3})$ indicated by the box with the hatched line in Fig.5. Then, we can determine 4 bytes of $(K^0_{3,0}, K^0_{3,1}, K^0_{3,2}\ K^0_{3,3})$, of K^0 by calculating the inverse of the key expansion routine. If the guessed 4 bytes are equal to 4 bytes of K^0 deduced in Section 7.2, we store the guessed values $(K^{13}_{0,0}, K^{13}_{0,1}, K^{13}_{0,2}, K^{13}_{0,3})$. Therefore, we can reduce the space of the key candidates of $(K^{13}_{0,0}, K^{13}_{0,1}, K^{13}_{0,2}, K^{13}_{0,3})$, by calculating 2^{32} times the inverse of the key expansion routine.

In the same way, we reduce the the space of the key candidates of $(K^{13}_{i,0}, K^{13}_{i,1}, K^{13}_{i,2}, K^{13}_{i,3})$ $(i = 1 \ldots 3)$.

7.4 Deduce the original 256-bit key

Using K_{14} deduced in Section 7.1 and the key candidates K^{13} deduced in Section 7.3, we calculate the key candidates of the original 256-bit key $k_{(256)}$ by applying the inverse calculation of the key expansion routine. If the candidates of the ciphertext calculated from $k_{(256)}$ is equal to the correct ciphertext, the key candidates of the original 256-bit key is correct and equal to the original key $K_{(256)}$.

8 Simulation Result

In order to verify the proposed attack and estimate the calculation time, we implement the attacks for 192-bit and 256-bit keys using the C code and execute it on a Xeon 3.0 GHz PC [3].

In the case of 192-bit key Using proposed attack method, the key candidate space of the right-half of K^{11} is reduced to 2^8 from 2^{64}. Therefore, an amount of calculation T needed to obtain the original key is calculated as follows.

$$\begin{aligned} T &= 2 \times 0.5 \times 2^{32} \times (\text{1R inverse calc.}) + 2^8 \times (\text{encryption}) \\ &= 2^{32} \times (\text{1R inverse calc.}) + 2^8 \times (\text{encryption}) \\ &\sim 2^{32} \times (\text{1R inverse calc.}) \end{aligned} \quad (6)$$

[3] This simulation was done with a single thread processing. Therefore, one core of the multi-core CPU is only used.

The notation R indicates the round. In the above calculation, the coefficient 2 of the first term means two kinds of calculation which is correct and faulty calculations. The coefficient 0.5 means that we calculate the half-state (8 bytes) by applying the 1 round inverse function.

In the result, we can obtain the original 192-bit key within 5 minutes.

In the case of 256-bit key Using proposed attack method, the space of the key candidates of $(K^{13}_{0,0}, K^{13}_{0,1}, K^{13}_{0,2}, K^{13}_{0,3})$ is reduced to 2^4 from 2^{32}. Moreover, the space of the key candidates of $(K^{13}_{i,0}, K^{13}_{i,1}, K^{13}_{i,2}, K^{13}_{i,3})$ $(i = 1 \ldots 3)$ is reduced to 2^3. Therefore, we need $2^{13} (= 2^4 \times 2^3 \times 2^3 \times 2^3)$ times encryption to determine the original 256-bit key.

Therefore, an amount of calculation T needed to obtain the original key is calculated as follows.

$$\begin{aligned} T &= 2^{32} \times (\text{key expansion}) + 2^{13} \times (\text{encryption}) \\ &\sim 2^{32} \times (\text{key expansion}) \end{aligned} \quad (7)$$

In the result, we can obtain the original 256-bit key within 10 minutes.

9 Conclusions

We propose the differential fault analysis on AES with 192 and 256-bit keys. In the proposed attack, we can obtain 192-bit key using 3 pairs of correct and faulty ciphertexts within 5 minutes. We can obtain 256-bit key using 2 pairs of correct and faulty ciphertexts and 2 pairs of correct and faulty plaintexts within 10 minutes. Both 192 and 256-bit keys are retrieved within a practical time.

References

[1] E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems," *Technion - Computer Science Department - Technical Report CS0901.revised - 1997*.

[2] H. Chen, W. Wu, and D. Feng, "Differential Fault Analysis on CLEFIA," In S. Qing, H. Imai, and G, Wang, editor, *Information and Communications Security*, vol.4861 of *Lecture Notes in Computer Science*, pp.284-295, Springer-Verlag, 2008.

[3] "An Improved Method of Differential Fault Analysis on the SMS4 Cryptosystem," In proceedings of *the First International Symposium on Data, Privacy, and E-Commerce 2007*, IEEE-CS, pp.175-180.

[4] J. Blömer and J.-P. Seifert, "Fault Based Cryptanalysis of the Advanced Encryption Standard (AES)," In R. N. Wright editor, *Financial Cryptography 2003 – FC 2003*, vol.2742 of *Lecture Notes in Computer Science*, pp.162-181, Springer-Verlag, 2003.

[5] P. Dusart, G. Letourneux and O. Vivolo, “Differential Fault Analysis on A.E.S.,” *Cryptology eprint Archive Report* 2003/010. Available at http://www.iacr.org/

[6] G. Piret and J. J. Quisquater, “A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD,” In C. D. Walter *et. al.* , editor, *Cryptographic Hardware and Embedded Systems – CHES 2003*, vol.2779 of *Lecture Notes in Computer Science*, pp.77-88, 2003.

[7] C. Giraud, “DFA on AES,” In H. Dobbertin, V. Rijmen and A. Sowa, editors, *Advanced Encryption Standard (AES): 4th International Conference, AES 2004*, vol.3373 of *Lecture Notes in Computer Science*, pp. 27-41, Springer-Verlag, 2005.

[8] A. Moradi, M. T. M. Shalmani and M. Salmasizadeh, “A Generalized Method of Differential Fault Attack Against AES Cryptosystem,” In L. Goubin and M. Matsui, editor, *Cryptographic Hardware and Embedded Systems – CHES 2006*, vol.4249 of *Lecture Notes in Computer Science*, pp.91-100, 2006.

[9] C.-N. Chen and S.-M. Yen, “Differential Fault Analysis on AES Key Schedule and Some Countermeasures,” *Australasian Conference on Information Security and Privacy 2003 (ACISP 2003)*, vol.2727 of *Lecture Notes in Computer Science*, pp.118-129, 2003.

[10] D. Peacham and B. Thomas, “A DFA attack against the AES key schedule,” SiVenture White Paper 001, 26 October 2006. Available at http://www.siventure.com/pdfs/AES_KeySchedule_DFA_whitepaper.pdf

[11] J. Takahashi, T. Fukunaga and K. Yamakoshi, “DFA Mechanism on the AES Key Schedule,” *Fault Diagnosis and Tolerance in Cryptography, FDTC2007*, IEEE-CS, pp.62–72.

[12] National Institute of Standards and Technology, *Advanced Encryption Standard*, **NIST FIPS PUB 197**, 2001.

[13] D. Mukhopadhyay, “An Improved Fault Based Attack of the Advanced Encryption Standard,” Africacrypt 2009, vol.5580 of LNCS, pp.421-434, 2009.

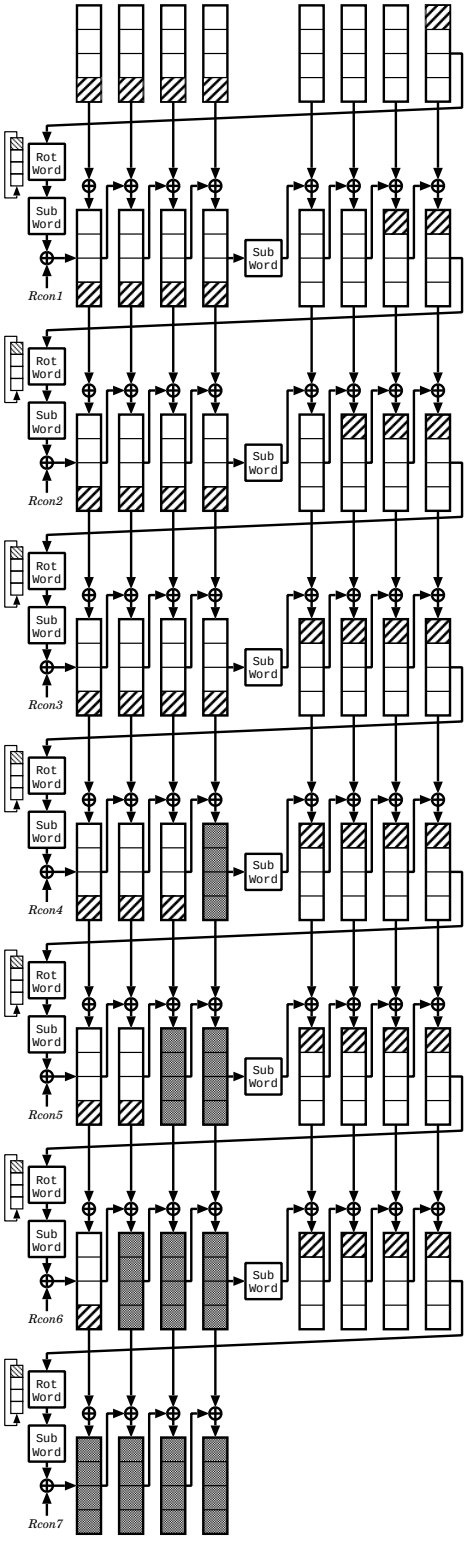

Figure 5. AES key expansion routine for 256-bit key.