Chapter 8
The Machines Take Over: Computer Cryptography

Abstract Modern cryptology rests on the shoulders of three men of rare talents. William Friedman, Lester Hill and Claude Shannon moved cryptology from an esoteric, mystical, strictly linguistic realm into the world of mathematics and statistics. Once Friedman, Hill, and Shannon placed cryptology on firm mathematical ground, other mathematicians and computer scientists developed the new algorithms to do digital encryption in the computer age. Despite some controversial flaws, the U.S. Federal Data Encryption Standard (DES) was the most widely used computer encryption algorithm in the 20th century. In 2001 a much stronger algorithm, the Advanced Encryption Standard (AES) that was vetted by a new burgeoning public cryptologic community, replaced it. This chapter introduces Hill and Shannon and explores the details of the DES and the AES.

8.1 The Shoulders of Giants

Modern cryptology rests on the shoulders of three giants of the 20th century. We've already talked about William F. Friedman and how his theoretical work, particularly the *Index of Coincidence*, brought statistics to cryptanalysis. Two other mathematicians made even more impressive impacts on cryptology in significantly different ways.

Lester S. Hill (1890–1961) was a mathematician who spent most of his career at Hunter College in New York City. In the June/July 1929 issue of *The American Mathematical Monthly* he published a paper titled *Cryptography in an Algebraic Alphabet* that marched cryptography a long way down the road towards being a mathematical discipline [7]. Hill's paper and its sequel in 1931 [8] were the first journal articles to apply abstract algebra to cryptography [9]. The substance of his paper was a new system of polygraphic encryption and decryption that used

J. F. Dooley, *A Brief History of Cryptology and Cryptographic Algorithms*, SpringerBriefs in Computer Science, DOI: 10.1007/978-3-319-01628-3_8,

invertible square matrices as the key elements and did all the arithmetic modulo 26. This is now known generally as matrix encryption, or the Hill cipher [3, p. 227]. The fundamental idea is to convert the letters of a message into numbers in the range 0 through 25 and to apply an invertible n × n square matrix to the numbers to create the ciphertext. The beauty of the system is that you can use as many of the letters of the message as you like and encrypt them all at once—a true polygraphic system. The system works by picking a size for the polygraphs, say 2. Then the user creates an invertible 2 × 2 matrix, M. The digraph letters are arranged as a two-row column vector (a 2 × 1 matrix) L and multiplying L by M creates the ciphertext. This looks like $M \cdot L = C$ where the · denotes matrix multiplication. Decryption just takes C and multiplies it by M^{-1} as in $M^{-1} \cdot C = L$. This system is easy to use but provides very good security. More importantly, Hill took another giant step in applying the tools of mathematics to cryptography.

The other mathematician we will discuss had the most significant and important impact on cryptology of the group. Claude Elwood Shannon (1916—2001) was both a mathematician and an electrical engineer and received his Ph.D. from M.I.T. in 1940. Two years earlier, his master's thesis was the first published work that linked Boolean algebra with electronic circuits—the basis of all modern computer arithmetic. In 1941 he joined the staff of Bell Telephone Laboratories and was soon working on communications and secrecy systems under contract from the War Department. In 1948 he was finally able to publish his work on communications systems as *A Mathematical Theory of Communication* [11], the foundational paper in information theory. In 1949 he followed with another seminal paper, *Mathematical Theory of Secrecy Systems* [12]. What Friedman had started and Hill continued, Shannon completed. In 60 dense pages *Secrecy Systems* placed cryptology on a firm mathematical foundation and provided the vocabulary and the theoretical basis for all the new cryptographic algorithms that would be developed over the next half-century. Shannon explored concepts like message entropy, language redundancy, perfect secrecy, what it means for a cipher system to be computationally secure, the unicity distance of a cipher system, the twin concepts of diffusion and confusion in cryptologic systems, product ciphers, and substitution-permutation networks.

Important for our discussion of computer algorithms are the concepts of *diffusion* and *confusion*. In general parlance, diffusion means spreading something widely across an area. A definition aptly used in Shannon's work. In Shannon's systems, messages are reduced to representations as numbers that are *binary digits* (bits) in a machine. A *secrecy system* is an algorithm that transforms a sequence of message bits into a different sequence of message bits. The idea of diffusion is to create a transformation that distributes the influence of each plaintext bit across a large number of ciphertext bits [3, p. 337]. Ideally the diffusion occurs across the entire ciphertext output. This is known as an *avalanche effect* because the effect of a single bit change is cascaded across many ciphertext bits. In a cipher, using transposition creates system diffusion. In diffusion the emphasis is on the relationship between the plaintext and the ciphertext. *Confusion* is the process of making the relationship between the plaintext and the ciphertext as complex as possible.

A cipher system does this via substitution [3, p. 337]. This complicates the transformation from plaintext to ciphertext, making the cryptanalyst's work much more difficult. In confusion the relationship is between the key bits and the ciphertext (a change in the key bits will change ciphertext bits). Shannon combined these two ideas into a *substitution-permutation network* (*S–P*) that uses diffusion and confusion to complicate the cipher. He also suggested that executing an S–P network a number of times—a *product cipher*—will also make the system that much more resistant to cryptanalysis.

8.2 Modern Computer Cipher Algorithms: The DES

Horst Feistel (1915–1990) struggled for many years to be allowed to do the cryptologic research he really wanted to do. But working for the government and government contractors made it difficult. When he finally started work at IBM's T.J. Watson Research Center in Yorktown, NY in the early 1970s he was finally able to do his cryptologic research. The result was a system called *Lucifer*. Lucifer was a very secure computer-based cipher system that IBM marketed and sold within the United States and—in a weakened version—abroad [6]. This was in response to the increasing amount of business being done via computer and the increasing number of financial transactions being handled across networks. Then, in 1973, the National Bureau of Standards put out a call for cryptographic algorithms that would be a federal standard and would be used to encrypt unclassified government data. It was clear that any algorithm that was a federal standard would also become very popular in the business world, so IBM submitted Lucifer as a candidate. It turned out that Lucifer was the only acceptable algorithm and it was adopted as Federal Information Processing Standard 46 (FIPS-46) on 15 July 1977 and renamed the federal Data Encryption Standard or DES [1].

8.2.1 How Does the DES Work?

The DES is a *symmetric block cipher algorithm*. It uses a single key to both encrypt and decrypt data (the symmetric part). It operates on data in 64-bit blocks (eight characters at a time), using a 56-bit key. It passes each block through the heart of the algorithm—a *round*—16 times before outputting the result as ciphertext. Each round breaks the 64-bit block into two 32-bit halves and then implements a substitution-permutation network using part of the key, called a sub-key, to produce an intermediate ciphertext that is then passed back again for the next round. Figure 8.1 diagrams the data flow of a round [1].

In more detail, the 64-bit input to DES is put through an *initial permutation* (IP) that rearranges the bits. The 64-bits are then divided into two halves, Left and Right and put through a round. In a round, nothing is done to the Right half.

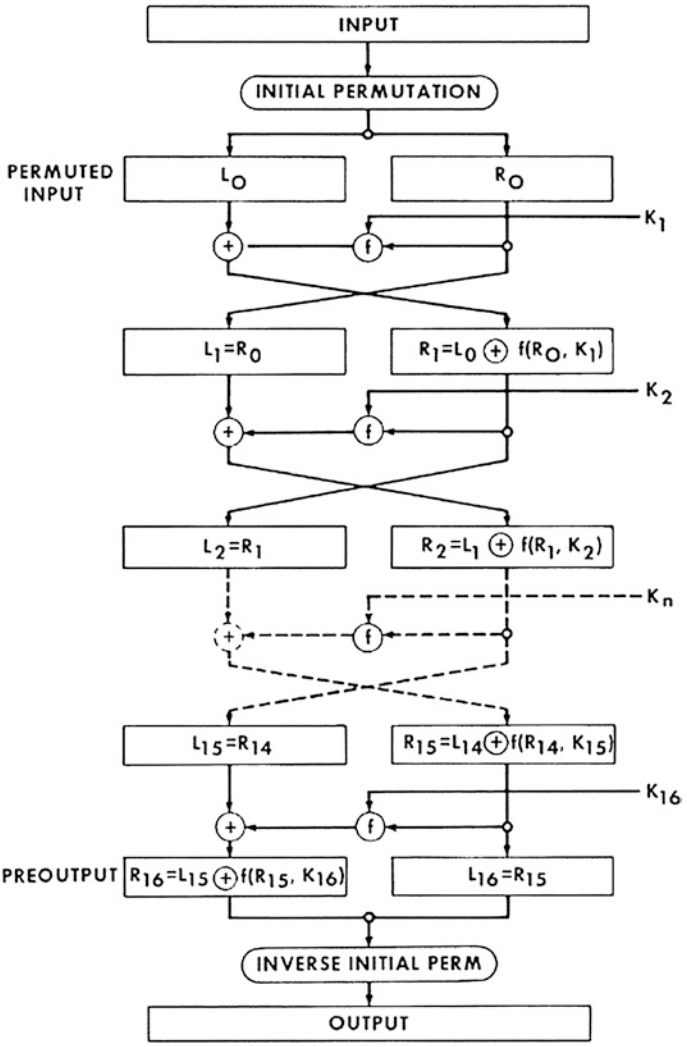

Fig. 8.1 Sixteen rounds of DES [1]

It becomes the Left half of the next round. The Right 32-bits are first put through a mixing function *f*(*Right*, *Key*) where Key is a sub-key generated by the *key scheduler*. The output of the function f() is exclusive-or'ed with the Left half. The result of this operation becomes the Right half input to the next round and the original Right half is the Left input to the next round. After the sixteenth round, the 64-bit output is put through a permutation that is the inverse of the initial permutation above. The resulting output is the 64-bit ciphertext.

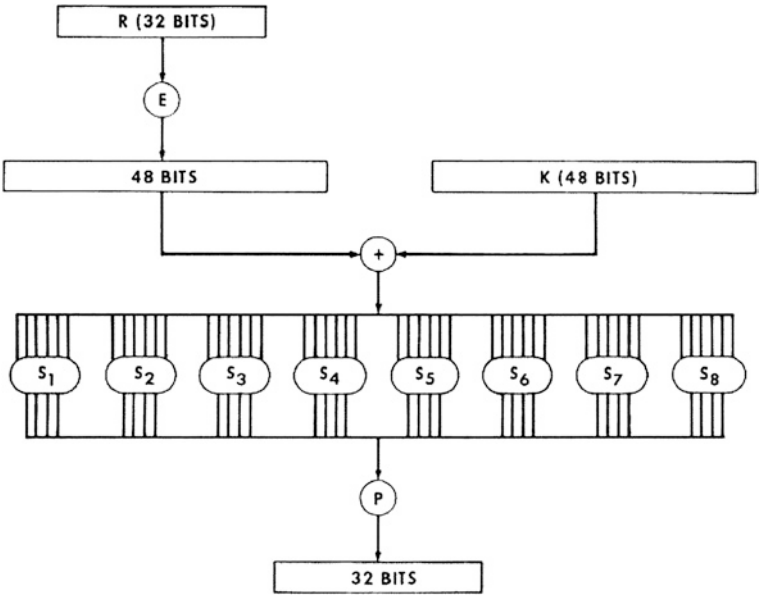

Fig. 8.2 Internals of the f() function [1]

8.2.2 The f() Function

The f() function takes as input the 32-bit right half of the input and a 48-bit sub-key generated by the key scheduler. This is illustrated in Fig. 8.2.

The first thing the f() function wants to do is XOR the right half with the sub-key. However, the generated sub-key is 48-bits and the right half of the data is only 32-bits. So the data must first go through the expansion block E. E performs a transformation that changes the right half into a 48-bit output using the expansion table in Fig. 8.3.

The 48-bit output of the exclusive or is broken up into 8 groups of 6 bits each and these 6-bit quantities are use as indexes into the substitution or S-boxes to select a four-bit output quantity. Block S_1 in Fig. 8.4 illustrates this selection.

The four-bit values from the 8 S-boxes are then combined and permuted one last time to make the 32-bit output of the function.

8.2.3 The Key Scheduler

The 56-bit DES key is broken up via the key scheduler into 48-bit sub-keys and a different sub-key is used for each round. The diagram for the key scheduler is in Fig. 8.5.

Fig. 8.3 E bit expansion table

E BIT-SELECTION TABLE

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

The original key is permuted and then broken up into two 28-bit halves. These halves are left shifted by an amount that depends on which round the key is destined for. The shift, though, is always either 1 or 2. The two 28-bit halves are then recombined, permuted, and 48-bits are selected for the round key. From the introduction of DES, ciphers that have this particular design of round are said to have *Feistel cipher structures* or *Feistel architectures*.

While the DES looks complicated, note that the only operations that are performed are XOR (exclusive or), bit shifting, and permutation of bits, all very simple operations in hardware. This allows DES to be fast.

8.2.4 Discussion of DES

With multiple substitutions and transpositions (disguised as permutations), the DES does a very good job of implementing Shannon's confusion and diffusion. It was not without controversy, though. Two particular areas stand out.

First, the key is too short [5, 10]. A 56-bit key only yields a key space of 2^{56} possible keys. This is only about 10^{18} or a quintillion keys, about half of which would need to be tried before the correct key was found to decrypt a message using brute force. Now this is not a small number, but with 201x computers we

S1

Row No.	Column Number															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Fig. 8.4 Substitution box S_1

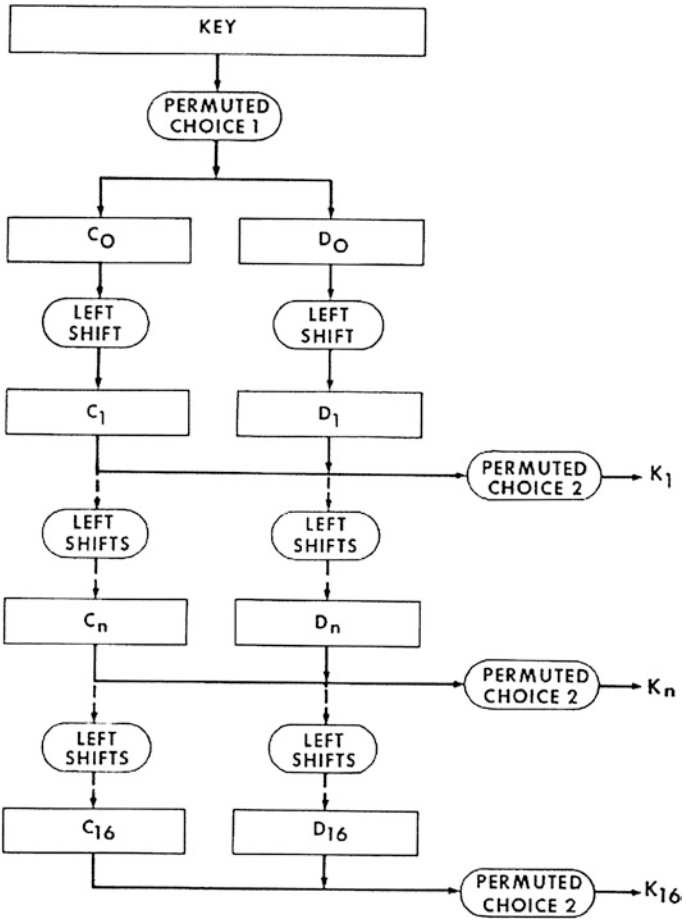

Fig. 8.5 The key scheduler for DES [1]

are talking less than a day to break a DES key. Even in the 1970s it was estimated that one could spend about $20 million dollars and create a special purpose machine that would break DES. In 1997 a network of thousands of computers on the Internet broke a DES key in a little over a month's time. And a year later, a special purpose computer built by the Electronic Frontier Foundation for less than $250,000 broke a DES key in less than three days [3, p. 385]. If a not-for-profit civil liberties organization can break a DES key in that short a time, surely a well-funded corporation or government can do it in less.

Why was the key so short? The original Lucifer key lengths were 64-bits and 128-bits, so why was the key shortened for DES? The prevailing theory at the time was that the NSA had requested the shorter key because their computing technology could break a 56-bit key in short order, but not anything larger.

The second piece of controversy is that at its introduction there was much complaint and discussion about the design of the substitution boxes of the DES. IBM and the NBS were closed-mouthed about how the particular values in each of the eight S-boxes were chosen and why [5, 10]. Again, suspicion fell on the NSA. This time the suspicion was that the design afforded the NSA a back door into the cipher. None of these accusations have been proven, and DES has stood up to heavy use for a quarter of a century. But by the mid-1990s it was beginning to show its age. Moore's law was making it more and more likely that cheap systems for breaking the DES would be available soon. So the National Institute of Science and Technology (the successor to the NBS) decided it was time for a new algorithm.

8.3 The Advanced Encryption Standard

In 1997 NIST sent out a call for potential successors for the DES. The climate was much different than in the early 1970s; by the 1990s there was a flourishing international community of researchers and practitioners in cryptology. Fifteen candidates were accepted and presented their algorithms at a NIST conference in 1998. By August 1999 the list was down to the top five candidates, RC6 from RSA, Inc in the U.S., MARS from IBM, Twofish from Counterpane in the U.S., Serpent from an English/Israeli/Danish group, and Rijndael from a group in Belgium. At this point all five algorithms were published and the international community was challenged to evaluate them and look for weaknesses. NIST also did its own evaluations.

In August 2000 Rijndael was chosen as the next standard and the new Advanced Encryption Standard (FIPS-197) was published in November 2001 [2].

AES is a symmetric key block cipher, just like DES. It uses a 128-bit input block, and gives the user three choices for key sizes, 128-bits, 192-bits, and 256-bits. The number of rounds varies depending on the key size. AES-128 uses 10 rounds, AES-192 uses 12 rounds, and AES-256 uses 14 rounds. The key data structure in AES is called *The State*. It is a 4×4 matrix of bytes (so 16 bytes * 8 bits/byte = 128-bits) that is acted upon by the algorithm to produce a 128-bit output. The basic algorithm for AES looks like Fig. 8.6.

In Fig. 8.6 N_b is the number of bytes in the input data block, and N_r is the number of rounds. Note that each round is basically four steps, *SubBytes*, *ShiftRows*, *MixColumns*, and *AddRoundKey*. The final round (outside the for loop) skips the MixColumns step.

Figures 8.7, 8.8, 8.9 and 8.10 illustrate each of these steps.

Notice that AES is not a Feistel architecture because it does not separate the input block into two halves; instead it is an iterative cipher, operating on the entire block in every round. It also is not invertible as written. To do decryption, you must apply the round structure in reverse. As with DES, AES provides

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[0, Nb-1])                  // See Sec. 5.1.4

   for round = 1 step 1 to Nr-1
      SubBytes(state)                              // See Sec. 5.1.1
      ShiftRows(state)                             // See Sec. 5.1.2
      MixColumns(state)                            // See Sec. 5.1.3
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   out = state
end
```

Fig. 8.6 The basic AES algorithm [2]

a key scheduler to create sub-keys, one for each round. The key scheduler is in Fig. 8.11.

Rijndael is designed to be easy to implement on architectures from 8-bit through at least 64-bit. Its operations can either be pre-computed or done using very simple operations. SubBytes just needs a table of 256 entries. ShiftRows is just simple byte shifting. MixColumns can also be implemented as a table look-up, and AddRoundKey just uses XOR.

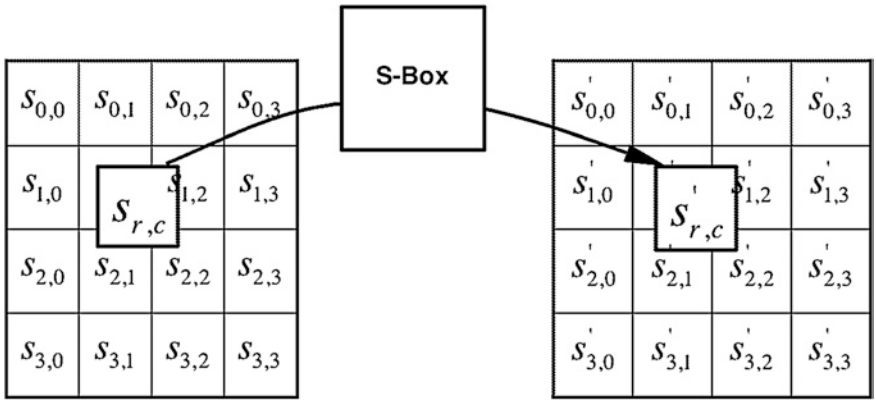

Fig. 8.7 The SubBytes substitution [2]

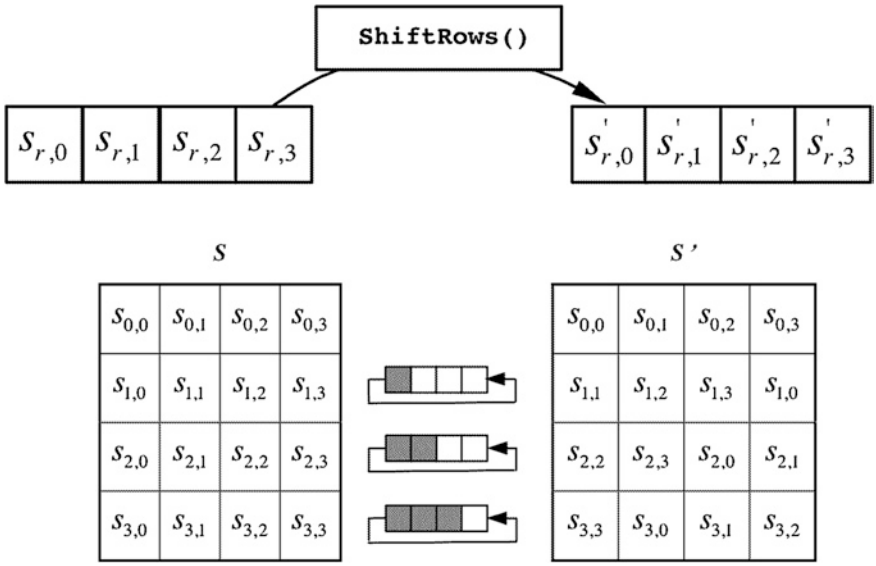

Fig. 8.8 ShiftRows function [2]

As opposed to DES, there has been no controversy with the adoption of Rijndael as the AES. This is because the entire process of picking the algorithm was open and transparent. After Rijndael was selected, the cryptographic community was given over a year to try to find flaws or weaknesses in the algorithm. The authors also published their own book on the design of the algorithm, providing their reasons for all their design decisions [4].

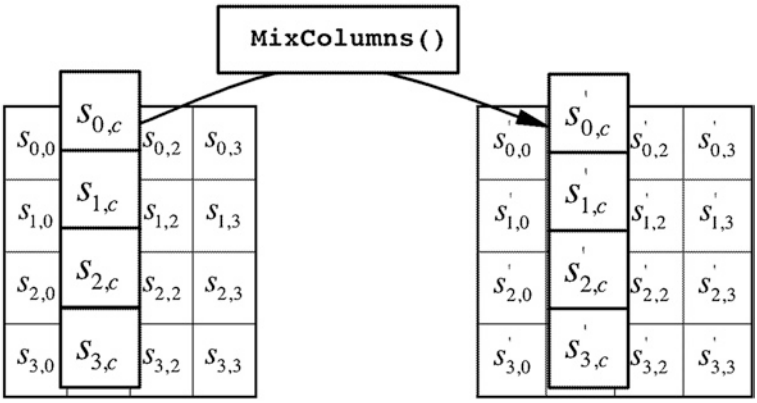

Fig. 8.9 The MixColumns function [2]

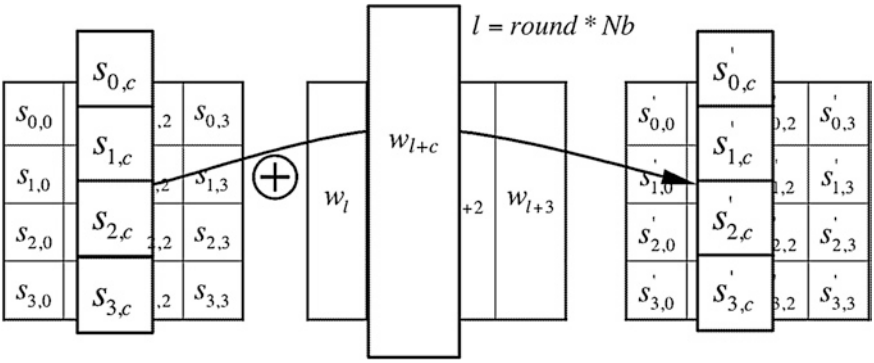

Fig. 8.10 The AddRoundKey function [2]

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

   i = 0

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

   i = Nk

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

Fig. 8.11 The AES key scheduler [2]

References

1. Anonymous. 1999. Data Encryption Standard (DES) FIPS 46-3, ed. National Institute of Standards and Technology. Gaithersburg, MD: United States Department of Commerce.
2. Anonymous. 2001. Federal Information Processing Standard 197: Advanced encryption standard (AES), ed. National Institute of Standards and Technology. Gaithersburg, MD: United States Department of Commerce.

3. Bauer, Craig P. 2013. *Secret history*: *The story of cryptology*. Boca Raton, FL: CRC Press.
4. Daemen, Joan, and Vincent Rijmen. 2002. *The design of Rijndael*: *AES—The advanced encryption standard*. New York: Springer.
5. Diffie, Whitfield, and Martin Hellman. 1977. Exhaustive cryptanalysis of the NBS data encryption standard. *IEEE Computer* 10(6): 74–84.
6. Feistel, Horst. 1973. Cryptography and computer privacy. *Scientific American* 228(5): 15–23.
7. Hill, Lester S. 1929. Cryptography in an algebraic alphabet. *The American Mathematical Monthly* 36: 306–312.
8. Hill, Lester S. 1931. Concerning certain linear transformation apparatus of cryptography. *The American Mathematical Monthly* 38: 135–154.
9. Kahn, David. 1967. *The codebreakers*: *The story of secret writing*. New York: Macmillan (Hardcover).
10. Morris, R., N.J.A. Sloane, and A.D. Wyner. 1977. Assessment of the National Bureau of standards proposed federal data encryption standard. *Cryptologia* 1(3): 281–291.
11. Shannon, Claude. 1948. A mathematical theory of communication, parts I and II. *Bell System Technical Journal* 27: 379–423, 623–656.
12. Shannon, Claude. 1949. Communication Theory of Secrecy Systems. *Bell System Technical Journal* 28(4): 656–715.