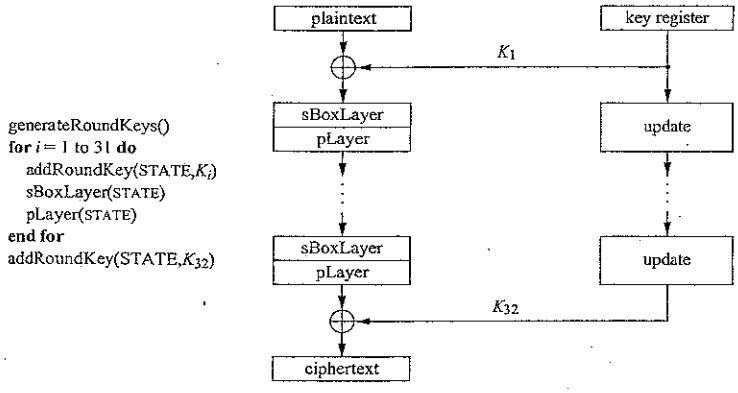
3.7.3 Lightweight Cipher PRESENT

Over the last few years, several new block algorithms which are classified as "lightweight ciphers" have been proposed. Lightweight commonly refers to algorithms with a very low implementation complexity, especially in hardware. Trivium (Sect. 2.3.3) is an example of a lightweight stream cipher. A promising block cipher candidate is *PRESENT*, which was designed specifically for applications such as RFID tags or other pervasive computing applications that are extremely power or cost constrained. (One of the book authors participated in the design of PRESENT.)




```
generateRoundKeys()
for i = 1 to 31 do
  addRoundKey(STATE,K_i)
  sBoxLayer(STATE)
  pLayer(STATE)
end for
addRoundKey(STATE,K_32)
```

Fig. 3.20 Internal structure and pseudocode of the block cipher PRESENT

Unlike DES, PRESENT is not based on a Feistel network. Instead it is a substitution-permutation network (SP-network) and consists of 31 rounds. The block length is 64 bits, and two key lengths of 80 and 128 bits are supported. Each of the 31 rounds consists of an XOR operation to introduce a round key K_i for $1 \leq i \leq 32$, where K_{32} is used after round 31, a nonlinear substitution layer (sBoxLayer) and a linear bitwise permutation (pLayer). The nonlinear layer uses a single 4-bit S-box S, which is applied 16 times in parallel in each round. The key schedule generates 32 round keys from the user supplied key. The encryption routine of the cipher is described in pseudocode in Fig. 3.20, and each stage is now specified in turn.

addRoundKey At the beginning of each round, the round key K_i is XORed to the current STATE.

sBoxlayer PRESENT uses a single 4-bit to 4-bit S-box. This is a direct consequence of the pursuit of hardware efficiency, since such an S-Box allows a much more compact implementation than, e.g., an 8-bit S-box. The S-box entries in hexadecimal notation are given in Table 3.14.

Table 3.14 The PRESENT S-box in hexadecimal notation

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

The 64 bit data path $b_{63} \ldots b_0$ is referred to as *state*. For the sBoxLayer the current state is considered as sixteen 4-bit words $w_{15} \ldots w_0$, where $w_i = b_{4*i+3} || b_{4*i+2} || b_{4*i+1} || b_{4*i}$ for $0 \leq i \leq 15$, and the output are the 16 words $S[w_i]$.

pLayer Just like DES, the mixing layer was chosen as a bit permutation, which can be implemented extremely compactly in hardware. The bit permutation used in PRESENT is given by Table 3.15. Bit i of STATE is moved to bit position $P(i)$.

Table 3.15 The permutation layer of PRESENT

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

The bit permutation is quite regular and can in fact be expressed in the following way:

$$P(i) = \begin{cases} i \cdot 16 \mod 63, & i \in \{0,\ldots,62\} \\ 63, & i = 63. \end{cases}$$

Key Schedule We describe in the following the key schedule for PRESENT with an 80-bit key. Since the main applications of PRESENT are low-cost systems, this key length is in most cases appropriate. (Details of the key schedule for PRESENT-128 can be found in [29].) The user-supplied key is stored in a key register K and is represented as $k_{79}k_{78}\ldots k_0$. At round i the 64-bit round key $K_i = \kappa_{63}\kappa_{62}\ldots\kappa_0$ consists of the 64 leftmost bits of the current contents of register K. Thus at round i we have:

$$K_i = \kappa_{63}\kappa_{62}\ldots\kappa_0 = k_{79}k_{78}\ldots k_{16}$$

The first subkey K_1 is a direct copy of 64 bit of the user supplied key. For the following subkeys $K_2,\ldots,K_{32}$ the key register $K = k_{79}k_{78}\ldots k_0$ is updated as follows:

1. $[k_{79}k_{78}\ldots k_1k_0] = [k_{18}k_{17}\ldots k_{20}k_{19}]$
2. $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$
3. $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus \texttt{round\_counter}$

Thus, the key schedule consists of three operations: (1) the key register is rotated by 61 bit positions to the left, (2) the leftmost four bits are passed through the PRESENT S-box, and (3) the `round_counter` value i is XORed with bits $k_{19}k_{18}k_{17}k_{16}k_{15}$ of K, where the least significant bit of `round_counter` is on the right. This counter is a simple integer which takes the values (00001, 00010, ... , 11111). Note that for the derivation of K_2 the counter value 00001 is used; for K_3, the counter value 00010; and so on.

Implementation As a result of the aggressively hardware-optimized design of PRESENT, its software performance is not very competitive relative to modern ciphers like AES. An optimized software implementation on a Pentium III CPU in C achieves a throughput of about 60 Mbit/s at a frequency of 1 GHz. However, it performs quite well on small microprocessors, which are common in inexpensive consumer products.

PRESENT-80 can be implemented in hardware with area requirements of approximately 1600 gate equivalences [147], where the encryption of one 64-bit plaintext block requires 32 clock cycles. As an example, at a clock rate of 1 MHz, which is quite typical on low-cost devices, a throughput of 2 Mbit/s is achieved, which is sufficient for most such applications. It is possible to realize the cipher with as few as approximately 1000 gate equivalences, where the encryption of one 64-bit plaintext requires 547 clock cycles. A fully pipelined implementation of PRESENT with 31 encryption stages achieves a throughput of 64 bit per clock cycle, which can be tranlsated into encryption throughputs of more than 50 Gbit/s.

Even though no attacks against PRESENT are known at the time of writing, it should be noted that it is a relatively new block cipher.

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F