COVER SHEET

Nomination of SAFER+ as Candidate Algorithm for the Advanced Encryption Standard (AES)

SUBMISSION DOCUMENT of 12 June 1998

- **Name of algorithm submitted:**
  SAFER+

- **Principal submitter:**
  Cylink Corporation
  910 Hermosa Court
  Sunnyvale, CA 94086
  (represented by Dr. Lily Chen
  Tel: (408) 735 5840; Fax: (408) 735 6645
  e-mail: lilychen@cylink.com)

- **Auxiliary submitters:**
  Prof. James L. Massey (Prof. emeritus, ETH Zurich, Switzerland)
  Prof. Gurgen H. Khachatrian (Academy of Sciences, Armenia)
  Dr. Melsik K. Kuregian (Academy of Sciences, Armenia)

- **Inventors of algorithm:**
  Prof. James L. Massey (Prof. emeritus, ETH Zurich, Switzerland)
  Prof. Gurgen H. Khachatrian (Academy of Sciences, Armenia)
  Dr. Melsik K. Kuregian (Academy of Sciences, Armenia)

- **Owner of algorithm:**
  Cylink relinquishes all proprietary rights to SAFER+ and consigns this algorithm to the public domain.

- **Signature of submitter:**

1 Introduction

This report nominates the cipher SAFER+ for use in the Advanced Encryption Standard (AES). The proposed cipher SAFER+ is based on the existing SAFER family of ciphers, which comprises the ciphers SAFER K-64, SAFER K-128, SAFER SK-64, SAFER SK-128, and SAFER SK-40. The block size of all the ciphers in the existing SAFER family is 64 bits, while the user-selected-key length is 40 or 64 or 128 bits as indicated in the name of the particular cipher.

The ciphers in the existing SAFER family are non-proprietary ciphers and were designed by Prof. James L. Massey of the ETH Zurich (Swiss Federal Institute of Technology, Zurich) at the request of Cylink Corporation. The first of these ciphers, SAFER K-64, was publicly announced at the Dec. 9–11, 1993, Fast Software Encryption workshop in Cambridge, England [1]. The other ciphers in the SAFER family differ from SAFER K-64 only in the key schedules that they use and in the recommended number of encryption rounds to be used. The name "SAFER" was originally chosen by Massey as an acronym for "Secure And Fast Encryption Routine".

The proposed cipher SAFER+ offers substantial improvements over the previous ciphers in the SAFER family, which is one of the grounds for choosing the name SAFER+ for this cipher. The name SAFER+ also serves to distinguish the proposed cipher from those in the existing SAFER family. The improvements incorporated in SAFER+ were developed by Massey together with Prof. Gurgen H. Khachatrian (Academy of Sciences, Armenia) and Dr. Melsik K. Kuregian (Academy of Sciences, Armenia). SAFER+ provides for a block size of 128 bits for the plaintext and ciphertext and accommodates three different user-selected-key lengths, namely 128, 192 and 256 bits.

This nomination is structured as follows. In Section 2, we provide a complete description of the SAFER+ encrypting, decrypting and key–schedule algorithms. In Section 3, we explain in detail the rationale for the design of SAFER+. Section 4 documents the computational efficiency of SAFER+ for both software and hardware implementations. The cryptographic strength of SAFER+ and certain analyses are described in Section 5. The advantages and limitations of SAFER+ are indicated in Section 6. The non-proprietary status of SAFER+ is asserted in Section 7. The final Section 8 lists the diskettes included with this nomination.

2 Specification of the SAFER+ Algorithm

2.1 System Specification of the SAFER+ Algorithm

The general encryption and decryption structure of the SAFER+ algorithm is shown in Fig. 1.

As indicated in Fig. 1, the input for encryption is the plaintext block of 16 bytes. (The convention used in this proposal for numbering bytes and bits is the same as that in the Data Encryption Standard [2], i.e., byte 1 is the most significant [or leftmost] byte in the block and byte 16 is the least significant [or rightmost] byte. Similarly, bit 1 is the most significant bit of a byte and bit 8 is the least significant bit.) The plaintext block then passes through r rounds of encryption where r is determined by the key length chosen for encryption in the following manner:

- if key length = 128 bits, then $r = 8$ rounds.
- if key length = 192 bits, then $r = 12$ rounds.
- if key length = 256 bits, then $r = 16$ rounds.

Two 16-byte round subkeys are used within each round of encryption. These round subkeys $(K_1, K_2, \ldots, K_{2r})$ are determined from the user-selected key K according to a key schedule described below. Finally, the last round subkey K_{2r+1} is "added" to the block produced by the r rounds of encryption in the manner that bytes 1, 4, 5, 8, 9, 12, 13, and 16 are added together bit-by-bit modulo two (the bitwise "exclusive-or" operation) while bytes 2, 3, 6, 7, 10, 11, 14 and 15 are added together modulo 256 ("byte addition"). This "addition" of round subkey K_{2r+1} constitutes the *output transformation* for SAFER+ encryption and produces the ciphertext block of 16 bytes.

As indicated in Fig. 1, the input for decryption is the ciphertext block of 16 bytes. Decryption begins with the *input transformation* that undoes the output transformation in the encryption process. In the input transformation, the round subkey K_{2r+1} is "subtracted" from the ciphertext block in the manner that the round subkey bytes 1, 4, 5, 8, 9, 12, 13, and 16 are added together bit-by-bit modulo two (the bitwise "exclusive-or" operation) to the corresponding ciphertext bytes (because modulo-two addition and subtraction coincide) while round subkey bytes 2, 3, 6, 7, 10, 11, 14 and 15 are subtracted modulo 256 ("byte subtraction") from the corresponding ciphertext bytes. The result of this "subtraction" is the same 16-byte block as was

produced from the r rounds of encryption before the output transformation was applied. This block then passes through the r rounds of decryption, round 1 of which undoes round r of encryption, round 2 of which undoes round $r-1$ of encryption, ... , and round r of which undoes round 1 of encryption to produce the original plaintext block. Note that the round keys for decryption are the same as those for encryption but are used in reverse order.

2.1.1 SAFER+ encryption round

The details of one round of encryption with SAFER+ are shown in Fig. 2. The first operation within round i, $1 \leq i \leq r$, is the "addition" of the round subkey K_{2i-1} to the 16-byte round input in the manner that bytes 1, 4, 5, 8, 9, 12, 13, and 16 are added together bit-by-bit modulo two while bytes 2, 3, 6, 7, 10, 11, 14 and 15 are added together modulo 256. The 16-byte result of this "addition" is then processed by a *nonlinear layer* in the manner that the value x of byte j is converted to 45^x mod 257 for bytes $j = 1$, 4, 5, 8, 9, 12, 13, and 16 (with the convention that when $x = 128$ then 45^{128} mod $257 = 256$ is represented by 0), while the value x of byte j is converted to $\log_{45}(x)$ for bytes $j = 2$, 3, 6, 7, 10, 11, 14 and 15 (with the convention that when $x = 0$ then the output $\log_{45}(0)$ is represented by 128). The round key K_{2i} is then "added" to the output of the nonlinear layer in the manner that bytes 2, 3, 6, 7, 10, 11, 14 and 15 are added together bit-by-bit modulo two, while bytes 1, 4, 5, 8, 9, 12, 13, and 16 are added together modulo 256. The 16-byte result of this "addition"

$$\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}]$$

is then postmultiplied by the matrix $\mathbf{M}$ modulo 256 to give the 16-byte round output

$$\mathbf{y} = [y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10}, y_{11}, y_{12}, y_{13}, y_{14}, y_{15}, y_{16}]$$

in the manner

$$\mathbf{y} = \mathbf{x}\mathbf{M}$$

where $\mathbf{M}$ is the 16×16 matrix

$$\begin{bmatrix}
2 & 2 & 1 & 1 & 16 & 8 & 2 & 1 & 4 & 2 & 4 & 2 & 1 & 1 & 4 & 4 \\
1 & 1 & 1 & 1 & 8 & 4 & 2 & 1 & 2 & 1 & 4 & 2 & 1 & 1 & 2 & 2 \\
1 & 1 & 4 & 4 & 2 & 1 & 4 & 2 & 4 & 2 & 16 & 8 & 2 & 2 & 1 & 1 \\
1 & 1 & 2 & 2 & 2 & 1 & 2 & 1 & 4 & 2 & 8 & 4 & 1 & 1 & 1 & 1 \\
4 & 4 & 2 & 1 & 4 & 2 & 4 & 2 & 16 & 8 & 1 & 1 & 1 & 1 & 2 & 2 \\
2 & 2 & 2 & 1 & 2 & 1 & 4 & 2 & 8 & 4 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 4 & 2 & 4 & 2 & 16 & 8 & 2 & 1 & 2 & 2 & 4 & 4 & 1 & 1 \\
1 & 1 & 2 & 1 & 4 & 2 & 8 & 4 & 2 & 1 & 1 & 1 & 2 & 2 & 1 & 1 \\
2 & 1 & 16 & 8 & 1 & 1 & 2 & 2 & 1 & 1 & 4 & 4 & 4 & 2 & 4 & 2 \\
2 & 1 & 8 & 4 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 4 & 2 & 2 & 1 \\
4 & 2 & 4 & 2 & 4 & 4 & 1 & 1 & 2 & 2 & 1 & 1 & 16 & 8 & 2 & 1 \\
2 & 1 & 4 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 8 & 4 & 2 & 1 \\
4 & 2 & 2 & 2 & 1 & 1 & 4 & 4 & 1 & 1 & 4 & 2 & 2 & 1 & 16 & 8 \\
4 & 2 & 1 & 1 & 1 & 1 & 2 & 2 & 1 & 1 & 2 & 1 & 2 & 1 & 8 & 4 \\
16 & 8 & 1 & 1 & 2 & 2 & 1 & 1 & 4 & 4 & 2 & 1 & 4 & 2 & 4 & 2 \\
8 & 4 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 1 & 2 & 1 & 4 & 2
\end{bmatrix}.$$

For instance, this operation gives

$$y_2 = 2x_1+x_2+x_3+x_4+4x_5+2x_6+x_7+x_8+x_9+x_{10}+2x_{11}+x_{12}+2x_{13}+2x_{14}+8x_{15}+4x_{16},$$

(where the arithmetic is modulo 256, i.e., normal byte arithmetic) as follows from the second column of the matrix $\mathbf{M}$.

2.1.2 SAFER+ decryption round

The details of one round of decryption with SAFER+ are shown in Fig. 3. The operations in the decryption round simply invert in reverse order the operations from the encryption round. Thus, the first operation in the decryption round is to postmultiply the 16-byte round input

$$\mathbf{y} = [y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10}, y_{11}, y_{12}, y_{13}, y_{14}, y_{15}, y_{16}]$$

by the the matrix $\mathbf{M}^{-1}$, which is the modulo 256 inverse of $\mathbf{M}$, to give the 16-byte result

$$\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}]$$

in the manner

$$\mathbf{x} = \mathbf{y}\mathbf{M}^{-1}.$$

The matrix $\mathbf{M}^{-1}$ is the 16×16 matrix

$$\begin{bmatrix}
2 & 254 & 1 & 254 & 1 & 255 & 4 & 248 & 2 & 252 & 1 & 255 & 1 & 254 & 1 & 255 \\
252 & 4 & 254 & 4 & 254 & 2 & 248 & 16 & 254 & 4 & 255 & 1 & 255 & 2 & 255 & 1 \\
1 & 254 & 1 & 255 & 2 & 252 & 1 & 255 & 1 & 255 & 1 & 254 & 2 & 254 & 4 & 248 \\
254 & 4 & 254 & 2 & 254 & 4 & 255 & 1 & 255 & 1 & 255 & 2 & 252 & 4 & 248 & 16 \\
1 & 255 & 2 & 252 & 1 & 255 & 1 & 254 & 1 & 254 & 1 & 255 & 4 & 248 & 2 & 254 \\
255 & 1 & 254 & 4 & 255 & 1 & 255 & 2 & 254 & 4 & 254 & 2 & 248 & 16 & 252 & 4 \\
2 & 252 & 1 & 255 & 1 & 254 & 1 & 255 & 2 & 254 & 4 & 248 & 1 & 255 & 1 & 254 \\
254 & 4 & 255 & 1 & 255 & 2 & 255 & 1 & 252 & 4 & 248 & 16 & 254 & 2 & 254 & 4 \\
1 & 255 & 1 & 254 & 1 & 255 & 2 & 252 & 4 & 248 & 2 & 254 & 1 & 254 & 1 & 255 \\
255 & 1 & 255 & 2 & 255 & 1 & 254 & 4 & 248 & 16 & 252 & 4 & 254 & 4 & 254 & 2 \\
1 & 254 & 1 & 255 & 4 & 248 & 2 & 254 & 1 & 255 & 1 & 254 & 1 & 255 & 2 & 252 \\
255 & 2 & 255 & 1 & 248 & 16 & 252 & 4 & 254 & 2 & 254 & 4 & 255 & 1 & 254 & 4 \\
4 & 248 & 2 & 254 & 1 & 254 & 1 & 255 & 1 & 254 & 1 & 255 & 2 & 252 & 1 & 255 \\
248 & 16 & 252 & 4 & 254 & 4 & 254 & 2 & 255 & 2 & 255 & 1 & 254 & 4 & 255 & 1 \\
1 & 255 & 4 & 248 & 2 & 254 & 1 & 254 & 1 & 255 & 2 & 252 & 1 & 255 & 1 & 254 \\
254 & 2 & 248 & 16 & 252 & 4 & 254 & 4 & 255 & 1 & 254 & 4 & 255 & 1 & 255 & 2
\end{bmatrix}.$$

Upon writing $-i$ to denote $256-i$ in modulo 256 arithmetic, the matrix $\mathbf{M}^{-1}$ can be written more simply as the 16×16 matrix

$$\begin{bmatrix}
2 & -2 & 1 & -2 & 1 & -1 & 4 & -8 & 2 & -4 & 1 & -1 & 1 & -2 & 1 & -1 \\
-4 & 4 & -2 & 4 & -2 & 2 & -8 & 16 & -2 & 4 & -1 & 1 & -1 & 2 & -1 & 1 \\
1 & -2 & 1 & -1 & 2 & -4 & 1 & -1 & 1 & -1 & 1 & -2 & 2 & -2 & 4 & -8 \\
-2 & 4 & -2 & 2 & -2 & 4 & -1 & 1 & -1 & 1 & -1 & 2 & -4 & 4 & -8 & 16 \\
1 & -1 & 2 & -4 & 1 & -1 & 1 & -2 & 1 & -2 & 1 & -1 & 4 & -8 & 2 & -2 \\
-1 & 1 & -2 & 4 & -1 & 1 & -1 & 2 & -2 & 4 & -2 & 2 & -8 & 16 & -4 & 4 \\
2 & -4 & 1 & -1 & 1 & -2 & 1 & -1 & 2 & -2 & 4 & -8 & 1 & -1 & 1 & -2 \\
-2 & 4 & -1 & 1 & -1 & 2 & -1 & 1 & -4 & 4 & -8 & 16 & -2 & 2 & -2 & 4 \\
1 & -1 & 1 & -2 & 1 & -1 & 2 & -4 & 4 & -8 & 2 & -2 & 1 & -2 & 1 & -1 \\
-1 & 1 & -1 & 2 & -1 & 1 & -2 & 4 & -8 & 16 & -4 & 4 & -2 & 4 & -2 & 2 \\
1 & -2 & 1 & -1 & 4 & -8 & 2 & -2 & 1 & -1 & 1 & -2 & 1 & -1 & 2 & -4 \\
-1 & 2 & -1 & 1 & -8 & 16 & -4 & 4 & -2 & 2 & -2 & 4 & -1 & 1 & -2 & 4 \\
4 & -8 & 2 & -2 & 1 & -2 & 1 & -1 & 1 & -2 & 1 & -1 & 2 & -4 & 1 & -1 \\
-8 & 16 & -4 & 4 & -2 & 4 & -2 & 2 & -1 & 2 & -1 & 1 & -2 & 4 & -1 & 1 \\
1 & -1 & 4 & -8 & 2 & -2 & 1 & -2 & 1 & -1 & 2 & -4 & 1 & -1 & 1 & -2 \\
-2 & 2 & -8 & 16 & -4 & 4 & -2 & 4 & -1 & 1 & -2 & 4 & -1 & 1 & -1 & 2
\end{bmatrix}.$$

For instance, this operation gives

$$x_3 = y_1 - 2y_2 + y_3 - 2y_4 + 2y_5 - 2y_6 + y_7 - y_8 + y_9 - y_{10} + y_{11} - y_{12} + 2y_{13} - 4y_{14} + 4y_{15} - 8y_{16},$$

(where the arithmetic is modulo 256, i.e., normal byte arithmetic) as follows from the third column of the matrix $\mathbf{M}^{-1}$.

The round subkey $K_{2r-2i+2}$ is then "subtracted" from $\mathbf{x}$ in the manner that the round subkey bytes 1, 4, 5, 8, 9, 12, 13, and 16 are subtracted modulo 256 from the corresponding bytes of $\mathbf{x}$ while round subkey bytes 2, 3, 6, 7, 10, 11, 14 and 15 are added bit-by-bit modulo 2 to the corresponding bytes of $\mathbf{x}$. The 16-byte result of this "subtraction" is then processed nonlinearly in the manner that the value x of byte j is converted to $\log_{45}(x)$ for bytes $j = 1$, 4, 5, 8, 9, 12, 13, and 16 (again with the convention that when $x = 0$ then $\log_{45}(0)$ is represented by 128), while the value x of byte j is converted to $45^x \bmod 257$ for bytes $j = 2$, 3, 6, 7, 10, 11, 14 and 15 (again with the convention that when $x = 128$ then $45^{128} \bmod 257 = 256$ is represented by 0). The round subkey $K_{2r-2i+1}$ is then "subtracted" from the 16-byte result in the manner that the round subkey bytes 1, 4, 5, 8, 9, 12, 13, and 16 are added bit-by-bit modulo 2 to the corresponding input bytes while round subkey bytes 2, 3, 6, 7, 10, 11, 14 and 15 are subtracted modulo 256 from the corresponding input bytes to produce the 16-byte round output.

2.1.3 SAFER+ Key Schedules

The $2r + 1$ 16-byte SAFER+ round subkeys required for the r rounds and for the output transformation of encryption (which are the same as those required for the input transformation and the r rounds of decryption) are produced from the input key according to a key schedule that depends on the key length selected.

Calculation of biases for key schedules

The key schedules of SAFER+ make use of 16-byte *bias words* to "randomize" the round subkeys produced. The required number of bias words is the same as the number $2r + 1$ of round subkeys, i.e., this number is 17, 25 or 33 depending on whether the user-selected-key length is 128 bits, 192 bits or 256 bits, respectively. The first bias word, however, is a "dummy" word that is never used but is convenient to have defined for programming purposes.

Let B_i denote the i-th bias word and let $B_{i,j}$ denote the j-th byte of this i-th bias word. For bias words B_2, B_3, ... B_{17}, which are used in all the key schedules and are the only bias words needed for a 128-bit user-selected key, the bias bytes are computed in the following manner:

$$B_{i,j} = 45^{(45^{17i+j} \bmod 257)} \bmod 257$$

(where $B_{i,j}$ is represented as 0 in case this expression gives a value of 256 and) where this expression applies for $i = 2, 3, \ldots, 17$ and $j = 1, 2, \ldots, 16$. The bias words B_{18}, B_{19}, ... B_{33}, of which only the first eight are needed for a 192-bit user-selected key but all sixteen of which are needed for a 256-bit user-selected key, are computed in the following manner:

$$B_{i,j} = 45^{17i+j} \bmod 257$$

(where $B_{i,j}$ is represented as 0 in case this expression gives a value of 256 and) where this expression applies for $i = 18, 19, \ldots, 33$ and $j = 1, 2, \ldots, 16$.

Table I gives lists all 32 bias words B_2, B_3, ... B_{33} that are used in the SAFER+ key schedules. These bias words were computed from the above formulas and are listed in the manner that the 16 bytes of B_2 form the first row, the 16 bytes of B_3 form the second row, ... , and the 16 bytes of B_{33} form the last row.

Key schedule for 128-bit user-selected key

The key schedule for the 128 bit (or 16 byte) input key is diagrammed in Fig. 4. The necessary 17 round subkeys for the 8 rounds and the output transformation of encryption are produced in the following manner. The user-selected key itself is used as the first round subkey K_1 and is also loaded into the first 16 byte positions of a 17-byte *key register*. The last byte position of this register is loaded with the bit-by-bit modulo-two sum of the 16 bytes of the user-selected key. Each byte of the key register is then rotated leftwards by 3 bit positions. The second round subkey K_2 is then computed as the modulo 256 sum of the bytes in the 16-byte bias word B_2 with the bytes in byte positions 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 and 17, respectively, of the key register. Each byte of the key register is then again rotated leftwards by 3 bit positions. The third round subkey K_3 is then computed as the modulo 256 sum of the bytes in the 16-byte bias word B_3 with the bytes in byte positions 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 and 1, respectively, of the key register. This processes continues with

70	151	177	186	163	183	16	10	197	55	179	201	90	40	172	100
236	171	170	198	103	149	88	13	248	154	246	110	102	220	5	61
138	195	216	137	106	233	54	73	67	191	235	212	150	155	104	160
93	87	146	31	213	113	92	187	34	193	190	123	188	153	99	148
42	97	184	52	50	25	253	251	23	64	230	81	29	65	68	143
221	4	128	222	231	49	214	127	1	162	247	57	218	111	35	202
58	208	28	209	48	62	18	161	205	15	224	168	175	130	89	44
125	173	178	239	194	135	206	117	6	19	2	144	79	46	114	51
192	141	207	169	129	226	196	39	47	108	122	159	82	225	21	56
252	32	66	199	8	228	9	85	94	140	20	118	96	255	223	215
250	11	33	0	26	249	166	185	232	158	98	76	217	145	80	210
24	180	7	132	234	91	164	200	14	203	72	105	75	78	156	53
69	77	84	229	37	60	12	74	139	63	204	167	219	107	174	244
45	243	124	109	157	181	38	116	242	147	83	176	240	17	237	131
182	3	22	115	59	30	142	112	189	134	27	71	126	36	86	241
136	70	151	177	186	163	183	16	10	197	55	179	201	90	40	172
220	134	119	215	166	17	251	244	186	146	145	100	131	241	51	239
44	181	178	43	136	209	153	203	140	132	29	20	129	151	113	202
163	139	87	60	130	196	82	92	28	232	160	4	180	133	74	246
84	182	223	12	26	142	222	224	57	252	32	155	36	78	169	152
171	242	96	208	108	234	250	199	217	0	212	31	110	67	188	236
137	254	122	93	73	201	50	194	249	154	248	109	22	219	89	150
233	205	230	70	66	143	10	193	204	185	101	176	210	198	172	30
98	41	46	14	116	80	2	90	195	37	123	138	42	91	240	6
71	111	112	157	126	16	206	18	39	213	76	79	214	121	48	104
117	125	228	237	128	106	144	55	162	94	118	170	197	127	61	175
229	25	97	253	77	124	183	11	238	173	75	34	245	231	115	35
200	5	225	102	221	179	88	105	99	86	15	161	49	149	23	7
40	1	45	226	147	190	69	21	174	120	3	135	164	184	56	207
8	103	9	148	235	38	168	107	189	24	52	27	187	191	114	247
53	72	156	81	47	59	85	227	192	159	216	211	243	141	177	255
62	220	134	119	215	166	17	251	244	186	146	145	100	131	241	51

Table I: Bias words B_2, B_3, ... B_{33} used in SAFER+ key schedules.

leftwards rotation by 3 bit positions of the key register followed by addition of the appropriate bias word to the sixteen bytes of the key register located one byte position rightwards (with position 1 understood to be to the right of position 17) of those previously used until the seventeenth round subkey K_{17} has been produced as the modulo 256 sum of the bytes in the 16-byte bias word B_{17} with the bytes in byte positions 17, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15, respectively, of the key register.

Key schedule for 192-bit user-selected key

The key schedule for the 192 bit (or 24 byte) input key is diagrammed in Fig. 5. The necessary 25 round subkeys for the 12 rounds and the output transformation of encryption are produced in the following manner. The first 16 bytes of the user-selected key itself are used as the first round subkey K_1 and the entire user-selected key is loaded into the first 24 byte positions of a 25-byte *key register*. The last byte position of this register is loaded with the bit-by-bit modulo-two sum of the 24 bytes of the user-selected key. Each byte of the key register is then rotated leftwards by 3 bit positions. The second round subkey K_2 is then computed as the modulo 256 sum of the bytes in the 16-byte bias word B_2 with the bytes in byte positions 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 and 17, respectively, of the key register. Each byte of the key register is then again rotated leftwards by 3 bit positions. The third round subkey K_3 is then computed as the modulo 256 sum of the bytes in the 16-byte bias word B_3 with the bytes in byte positions 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 and 18, respectively, of the key register. This processes continues with leftwards rotation by 3 bit positions of the key register followed by addition of the appropriate bias word to the sixteen bytes of the key register located one byte position rightwards (with position 1 understood to be to the right of position 25) of those previously used until the twenty-fifth round subkey K_{25} has been produced as the modulo 256 sum of the bytes in the 16-byte bias word B_{25} with the bytes in byte positions 25, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15, respectively, of the key register.

Key schedule for 256-bit user-selected key

The key schedule for the 256 bit (or 32 byte) input key is diagrammed in Fig. 6. The necessary 33 round subkeys for the 16 rounds and the output

transformation of encryption are produced in the following manner. The first 16 bytes of the user-selected key itself are used as the first round subkey K_1 and the entire user-selected key is loaded into the first 32 byte positions of a 33-byte *key register*. The last byte position of this register is loaded with the bit-by-bit modulo-two sum of the 32 bytes of the user-selected key. Each byte of the key register is then rotated leftwards by 3 bit positions. The second round subkey K_2 is then computed as the modulo 256 sum of the bytes in the 16-byte bias word B_2 with the bytes in byte positions 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 and 17, respectively, of the key register. Each byte of the key register is then again rotated leftwards by 3 bit positions. The third round subkey K_3 is then computed as the modulo 256 sum of the bytes in the 16-byte bias word B_3 with the bytes in byte positions 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 and 18, respectively, of the key register. This processes continues with leftwards rotation by 3 bit positions of the key register followed by addition of the appropriate bias word to the sixteen bytes of the key register located one byte position rightwards (with position 1 understood to be to the right of position 33) of those previously used until the thirty-third round subkey K_{33} has been produced as the modulo 256 sum of the bytes in the 16-byte bias word B_{33} with the bytes in byte positions 33, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15, respectively, of the key register.

2.2 Examples for SAFER+

The above description of SAFER+ is intended to be detailed and clear enough for a designer to program this cipher or implement it in silicon. The data in the following three examples, one for each key length accomodated by SAFER+, may he helpful to a designer who implements SAFER+ from this description.

Example for 128 bit key:

This is the 16 byte user-selected input key:

41 35 190 132 225 108 214 174 82 144 73 241 241 187 233 235

This is the plaintext block input:

179 166 219 60 135 12 62 153 36 94 13 28 6 183 71 222

This is the ciphertext block output:

224 31 182 10 12 255 84 70 127 13 89 249 9 57 165 220

These are the round subkeys K_1, K_2, ... ,K_{17}(one key per row):

41	35	190	132	225	108	214	174	82	144	73	241	241	187	233	235
95	140	213	201	6	109	133	156	73	129	66	88	55	119	11	35
155	204	34	225	28	64	236	49	74	22	114	92	224	214	2	135
147	134	176	54	199	141	87	219	38	162	98	167	109	138	186	230
123	29	255	9	250	122	240	218	65	124	92	57	59	43	149	127
96	204	15	93	122	189	245	243	244	52	219	76	177	210	163	209
56	190	201	32	12	248	157	109	168	81	214	221	102	105	53	81
15	26	46	250	110	124	137	222	74	13	5	12	134	18	149	185
207	61	251	224	179	66	183	96	253	60	37	78	211	15	222	9
68	215	94	56	94	49	35	230	120	133	111	195	97	68	203	173
78	156	190	181	130	222	6	159	38	59	53	238	123	180	138	107
221	238	152	211	241	232	248	255	101	167	37	36	134	238	244	243
55	111	165	66	105	237	214	179	86	233	14	214	53	115	165	201
34	65	73	224	185	205	107	140	123	117	55	254	4	179	82	236
212	162	91	17	41	175	56	251	163	238	13	249	50	54	180	74
51	1	59	215	18	174	202	253	151	91	101	89	167	98	148	104
127	111	186	111	62	132	35	230	184	23	199	252	186	75	227	149

Example for 192 bit key:

This is the 24 byte user-selected input key:

72 211 143 117 230 217 29 42 229 192 247 43 120 129 135 68
14 95 80 0 212 97 141 190

This is the plaintext block input:

123 5 21 7 59 51 130 31 24 112 146 218 100 84 206 177

This is the ciphertext block output:

92 136 4 63 57 95 100 0 150 130 130 16 193 111 219 133

These are the round subkeys K_1, K_2, ... ,K_{25}(one key per row):

72	211	143	117	230	217	29	42	229	192	247	43	120	129	135	68
228	19	92	241	113	159	97	57	203	246	12	140	102	100	206	212
207	8	99	60	174	31	209	61	245	100	20	206	71	237	136	20
116	144	139	195	190	180	183	56	153	175	238	227	30	183	38	64
203	244	99	193	51	125	219	109	169	217	54	191	156	142	104	148
22	239	205	38	146	20	146	55	215	3	8	88	204	105	68	249
81	172	23	225	198	221	183	133	31	179	47	182	27	111	118	79
127	140	52	207	149	77	66	145	85	208	203	178	175	28	133	221
98	109	169	26	58	8	85	185	20	114	82	144	35	143	255	241
198	76	40	108	141	30	230	151	41	238	122	69	93	77	10	161
249	234	96	39	233	245	140	44	114	140	73	206	195	174	42	233
80	251	36	15	162	21	100	89	232	71	36	103	86	235	224	121
159	204	127	200	202	80	169	200	91	225	32	84	29	210	217	45
5	16	118	236	212	100	12	180	59	5	43	61	255	84	117	174
75	4	180	234	222	181	121	249	40	141	7	209	63	79	194	30
62	196	1	125	59	184	186	33	148	43	36	193	111	210	50	44
150	165	231	177	142	4	68	206	55	13	10	66	62	64	1	201
214	8	119	125	177	125	240	93	252	48	13	15	186	191	27	64
64	181	231	131	235	128	228	221	128	103	122	205	247	222	251	67
163	52	25	87	255	30	226	3	59	210	109	183	238	217	21	119
161	204	183	247	236	18	27	216	144	106	189	108	198	172	181	23
91	184	191	102	144	211	193	129	76	236	98	52	96	163	183	129
191	248	46	126	152	7	7	93	96	14	160	4	25	186	5	119
192	114	239	192	51	61	230	252	111	254	33	200	208	43	187	78
143	113	1	157	233	54	219	119	237	10	59	129	85	211	113	141

Example for 256 bit key:

This is the 32 byte user-selected input key:

243	168	141	254	190	242	235	113	255	160	208	59	117	6	140	126
135	120	115	77	208	190	130	190	219	194	70	65	43	140	250	48

This is the plaintext block input:

127	112	240	167	84	134	50	149	170	91	104	19	11	230	252	245

This is the ciphertext block output:

88	11	25	36	172	229	202	213	170	65	105	153	220	104	153	138

These are the round subkeys K_1, K_2, ... ,K_{33}(one key per row):

243	168	141	254	190	242	235	113	255	160	208	59	117	6	140	126
139	3	168	175	58	22	155	9	202	189	140	116	138	140	159	160
79	106	89	130	97	241	87	53	44	104	83	239	137	123	230	91
135	64	189	96	76	232	119	234	185	169	247	237	146	170	88	134
72	134	80	54	212	123	105	110	121	33	134	98	52	32	154	104
163	86	112	51	130	129	154	181	26	134	37	20	89	250	234	247
140	201	127	96	42	29	171	151	51	155	21	26	167	164	102	196
104	207	48	235	151	236	210	50	156	255	239	22	88	156	48	124
124	77	130	42	55	141	90	243	141	139	117	221	31	236	244	241
197	19	168	84	177	70	183	99	242	7	228	37	71	245	10	22
48	238	159	72	43	131	234	115	58	223	72	37	0	174	213	135
112	245	45	25	22	8	150	159	130	63	223	81	86	72	213	94
111	20	207	107	98	226	219	156	27	182	112	84	8	122	0	73
72	147	147	168	97	245	178	178	234	128	43	148	60	142	78	137
95	236	154	78	106	234	105	110	252	141	194	187	9	22	153	181
133	243	37	225	228	56	101	192	148	1	115	15	166	137	231	80
15	190	10	254	138	97	57	206	229	135	125	244	244	230	34	220
159	33	225	93	155	37	240	210	208	196	155	189	231	200	180	72
8	8	230	218	40	128	143	123	29	212	231	55	63	163	59	198
61	44	212	65	255	123	215	232	158	62	185	249	20	219	49	71
97	161	7	247	215	186	66	244	235	196	207	158	214	141	51	112
10	51	191	189	205	13	154	92	31	125	236	180	103	151	130	107
147	248	233	104	98	206	222	244	228	90	164	60	184	17	84	144
192	72	62	14	106	244	155	32	210	30	227	197	131	165	131	124
61	235	116	79	159	220	252	138	238	24	35	23	40	25	226	241

93	161	122	246	226	231	79	107	198	26	184	70	203	16	143	243
6	205	174	16	62	118	90	51	204	193	53	89	129	121	153	174
103	111	122	242	173	210	158	92	9	170	200	7	204	201	114	100
122	205	144	105	143	242	226	65	82	65	62	95	72	148	33	20
110	126	69	119	140	18	11	148	13	241	248	63	163	8	160	108
243	39	181	99	141	92	163	101	136	199	249	26	61	2	94	204
59	173	26	102	224	26	44	65	61	205	215	231	13	244	95	191
105	207	46	4	213	100	3	230	101	185	50	97	159	248	247	191

3 Design Rationale

The major design principles underlying the SAFER+ algorithm are the following:

- Encrypting Structure
- Byte Orientation
- Group Operation at Round Input
- Use of Two Additive Groups
- Use of the Exponential and Logarithm in Nonlinear Layer
- Fast Diffusion via the Matrix **M**
- Scalability
- Biases in Key Schedule
- Parity Word and Selections in Key Schedule
- Number of Rounds

Each of these principles will now be explained.

Encrypting Structure

It should be emphasized that SAFER+, unlike the Date Encryption Algorithm (DEA) of DES and many later iterative block ciphers, is **not** a "Feistel" cipher. In a Feistel cipher, the round input is split into its left and right halves, say L_{i-1} and R_{i-1} for the input to round i. Some nonlinear function f from a half-block/round-key pair to a half block is then applied to the right half input R_{i-1} and the round key K_i. The value of this function is then added bit-by-bit modulo 2 to the left half input L_{i-1} to create a "new" left half $L_{i-1} \oplus f(R_{i-1}, K_i)$. The left and right halves are then "swapped" to produce the left half L_i and right half R_i of the output of round i as $L_i = R_{i-1}$ and $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$. The Feistel structure has the feature that the round function is invertible regardless of the choice of the function f. The "swapping of halves" is omitted in the final round of encryption with the result that decryption can be performed by using the encrypting

algorithm together with the round keys taken in reverse order. This "encryption/decryption similarity" is an implementation advantage for a Feistel cipher. On the other hand, diffusion in a Feistel cipher tends to be somewhat slow. For instance, it takes two rounds before every bit of the input can be altered by the algorithm. Moreover, the design of the function f, which is the main source of cryptographic strength, poses substantial problems that are generally solved by the use of "random-looking" tables that are difficult to justify analytically to a suspicious cryptanalyst.

A generalization of the Feistel cipher is the kind of iterative block cipher that is sometimes called a "substitution–permutation" cipher. Within a round of such a cipher, an invertible function controlled by the round key is first applied to the round input, then a permutation of coordinates is applied to the result. The substitution–permutation structure gives the designer more freedom than does the Feistel structure so that more rapid diffusion can be obtained, but in general the property of encryption/decryption similarity is lost, i.e., the encrypting and decrypting algorithms differ in more than the ordering of the round keys.

SAFER+ (as is also the case with all prior ciphers in the SAFER family) is neither a Feistel cipher nor a substitution–permutation cipher. There is no fundamental reason to alternate between substitutions and permutations (of coordinates) to create good confusion and diffusion. SAFER+ can be called a *substitution/linear-transformation* cipher, by which we mean that an invertible function controlled by the round key is first applied to the round input, then an invertible linear transformation is applied to the result. Because a permutation of coordinates is also an invertible linear transformation, the substitution/linear-transformation structure is a generalization of the substitution–permutation structure and offers the designer still more freedom. In particular, as will be seen below, extremely fast diffusion can be achieved by the judicious choice of the invertible linear transformation of the cipher.

Byte Orientation

SAFER+ is a *byte–oriented* cipher (as are also all prior ciphers in the SAFER family) in the sense that during encryption, decryption, and execution of the key schedule, only functions from one byte to one byte are used. This has the advantage that SAFER+ implements particularly well on 8-bit microprocessors. During set–up of the SAFER+ algorithm (i.e., during preparation of the tables for exponentiation of 45 modulo 257, for logarithms to the base

45, and for the key biases), some simple integer operations are performed, but this is done once and for all.

Group Operation at Round Input

As can be seen from Figs. 1 and 2, the first operation within the SAFER+ encrypting algorithm is to combine the 16-byte plaintext block with the 16-byte round subkey K_1 by the "addition" operation described in Section 2.1. This "addition" operation is a group operation. Moreover, the 16 bytes of round subkey K_1 are the first 16 bytes of the user–selected key regardless of the keylength chosen. If the user selects the key uniformly at random, then, for an arbitrarily chosen plaintext block, the result of the group operation is equally likely to be any of the 2^{128} possible 16 byte blocks. This means that the result of this group operation is statistically independent of the plaintext block and hence that this group operation provides *perfect secrecy* in the sense of Shannon [3] when the key is used only one time. Of course, the SAFER+ key will typically be used to encrypt very many plaintext blocks before being changed, but it is nonetheless worthwhile to incorporate this element of provable perfect secrecy into an encryption algorithm.

Use of Two Additive Groups

The group operation used to insert round subkeys as described in Section 2.1.1 is the operation of a "product group" consisting of 16 smaller groups. For 8 of these smaller groups, the operation is addition modulo 256 (i.e., usual byte addition) and for the remaining 8 smaller groups, the operation is bit-by-bit addition modulo 2 (i.e., the bitwise exclusive-or operation).

Using two different group operations to insert a round subkey permits enhanced "confusion" to be created by SAFER+ encryption. Consider for instance the function exptab(.) such that $\text{exptab}(x) = 45^x \bmod 257$ (with the convention that when $x = 128$ then $45^{128} \bmod 257 = 256$ is represented by 0) for $0 \leq x \leq 255$, which is the function corresponding to "exp" in Fig. 2. For this function,

$$\text{exptab}(x_1 + x_2) = \text{exptab}(x_1) \times \text{exptab}(x_2)$$

where "+" denotes addition modulo 256 and "×" denotes multiplication modulo 257. However, in general

$$\text{exptab}(x_1 \oplus x_2) \neq \text{exptab}(x_1) \times \text{exptab}(x_2)$$

where "$\oplus$" denotes bit-by-bit addition modulo 2 and "$\times$" denotes multiplication modulo 257. This difference is important–it shows that adding two bytes modulo 256 then evaluating exptab(.) of the result is equivalent to evaluating exptab(.) of each byte separately then multiplying the results, but this "morphism" property does not hold when the addition is bit-by-bit modulo 2. This is the reason that, in Fig. 2, those bytes of round subkey K_{2i-1} that are affected by the "exp" operation are inserted by bit-by-bit addition modulo 2 and *not* by addition modulo 256 as are the other bytes of K_{2i-1}. The insertion of those other bytes by addition modulo 256, however, is important for the strength of SAFER+ against linear cryptanalysis as will be discussed in Section 5.3. Having both types of addition available in SAFER+ makes it possible to exploit the particular strengths of each.

Use of the Exponential and Logarithm in Nonlinear Layer

As was described in Section 2.1.1 above, the 16 bytes within a round of SAFER+ that result from the "addition" of the first of the two round subkeys are used as inputs to either the function exptab(.) or its inverse logtab(y) = $\log_{45}(y)$ for $0 \leq y \leq 255$ (with the convention that $\log_{45}(0) = 128$), which is the function corresponding to "log" in Fig. 2. The choice of exptab(.) and logtab(.) as the mutually inverse nonlinear functions within the "nonlinear layer" of a round of SAFER+ was motivated by several factors. First of all, these are well-defined mathematical functions and their use obviates the suspicions of intentional weakness that might be raised if mutually inverse nonlinear functions defined only by "random looking" tables were chosen. Further, it was shown in [4] that, for the boolean functions determining single bits of exptab(.) and logtab(.), the number of terms of a given nonlinear order in the algebraic normal form of the function follows virtually the same distribution as for a randomly chosen boolean function. Moreover, Vaudenay [5], as a result of his cryptanalysis of SAFER K–64, concluded that the "choice [of exptab(.) and logtab(.) as the mutually inverse nonlinear functions] is a very good one" because a substantial fraction of all mutually inverse nonlinear functions when used in their place would lead to a known plaintext attack faster than exhaustive search.

Fast Diffusion via the Matrix M

In the previous ciphers of the SAFER family, for which the block size is always 8 bytes, the linear layer implements what Massey [1] has called the 3-dimensional Pseudo Hadamard Transform (PHT). The 3-dimensional PHT

corresponds to postmultiplication modulo 256 of the 8 byte input by the matrix

$$\mathbf{H_3} = \begin{bmatrix} 8 & 4 & 4 & 2 & 4 & 2 & 2 & 1 \\ 4 & 2 & 4 & 2 & 2 & 1 & 2 & 1 \\ 4 & 2 & 2 & 1 & 4 & 2 & 2 & 1 \\ 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 \\ 4 & 4 & 2 & 2 & 2 & 2 & 1 & 1 \\ 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 & 2 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

This matrix yields good diffusion in the sense that inputs with a small number of non-zero bytes generally produce outputs with many non-zero bytes. Note, however, that the input block [128 0 0 0 0 0 0 0] produces the output block [0 0 0 0 0 0 0 128] that also has only one non-zero byte.

For a block size of 16 bytes, one could use the 4-dimensional PHT, which corresponds to postmultiplication modulo 256 of the 16-byte input by the matrix

$$\mathbf{H_4} = \begin{bmatrix} 16 & 8 & 8 & 4 & 8 & 4 & 4 & 2 & 8 & 4 & 4 & 2 & 4 & 2 & 2 & 1 \\ 8 & 4 & 8 & 4 & 4 & 2 & 4 & 2 & 4 & 2 & 4 & 2 & 2 & 1 & 2 & 1 \\ 8 & 4 & 4 & 2 & 8 & 4 & 4 & 2 & 4 & 2 & 2 & 1 & 4 & 2 & 2 & 1 \\ 4 & 2 & 4 & 2 & 4 & 2 & 4 & 2 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 \\ 8 & 8 & 4 & 4 & 4 & 4 & 2 & 2 & 4 & 4 & 2 & 2 & 2 & 2 & 1 & 1 \\ 4 & 4 & 4 & 4 & 2 & 2 & 2 & 1 & 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 \\ 4 & 4 & 2 & 2 & 4 & 4 & 2 & 1 & 2 & 2 & 1 & 1 & 2 & 2 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 8 & 4 & 4 & 2 & 4 & 2 & 2 & 1 & 8 & 4 & 4 & 2 & 4 & 2 & 2 & 1 \\ 4 & 2 & 4 & 2 & 2 & 1 & 2 & 1 & 4 & 2 & 4 & 2 & 2 & 1 & 2 & 1 \\ 4 & 2 & 2 & 1 & 4 & 2 & 2 & 1 & 4 & 2 & 2 & 1 & 4 & 2 & 2 & 1 \\ 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 \\ 4 & 4 & 2 & 2 & 2 & 2 & 1 & 1 & 4 & 4 & 2 & 2 & 2 & 2 & 1 & 1 \\ 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 & 2 & 2 & 1 & 1 & 2 & 2 & 1 & 1 & 2 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Indeed this choice of the invertible linear transformation yields rather good diffusion. Note again, however, that the input block [128 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] produces the output block [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 128] that also has only one non-zero byte.

One of the major improvements of SAFER+ over the previous ciphers in the SAFER family is the choice of an invertible linear transformation that is considerably better than the PHT as described above. The matrix of this transformation, which was already given in Section 2.1.1, is

$$\mathbf{M} = \begin{bmatrix} 2 & 2 & 1 & 1 & 16 & 8 & 2 & 1 & 4 & 2 & 4 & 2 & 1 & 1 & 4 & 4 \\ 1 & 1 & 1 & 1 & 8 & 4 & 2 & 1 & 2 & 1 & 4 & 2 & 1 & 1 & 2 & 2 \\ 1 & 1 & 4 & 4 & 2 & 1 & 4 & 2 & 4 & 2 & 16 & 8 & 2 & 2 & 1 & 1 \\ 1 & 1 & 2 & 2 & 2 & 1 & 2 & 1 & 4 & 2 & 8 & 4 & 1 & 1 & 1 & 1 \\ 4 & 4 & 2 & 1 & 4 & 2 & 4 & 2 & 16 & 8 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 2 & 1 & 2 & 1 & 4 & 2 & 8 & 4 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 4 & 2 & 4 & 2 & 16 & 8 & 2 & 1 & 2 & 2 & 4 & 4 & 1 & 1 \\ 1 & 1 & 2 & 1 & 4 & 2 & 8 & 4 & 2 & 1 & 1 & 1 & 2 & 2 & 1 & 1 \\ 2 & 1 & 16 & 8 & 1 & 1 & 2 & 2 & 1 & 1 & 4 & 4 & 4 & 2 & 4 & 2 \\ 2 & 1 & 8 & 4 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 4 & 2 & 2 & 1 \\ 4 & 2 & 4 & 2 & 4 & 4 & 1 & 1 & 2 & 2 & 1 & 1 & 16 & 8 & 2 & 1 \\ 2 & 1 & 4 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 8 & 4 & 2 & 1 \\ 4 & 2 & 2 & 2 & 1 & 1 & 4 & 4 & 1 & 1 & 4 & 2 & 2 & 1 & 16 & 8 \\ 4 & 2 & 1 & 1 & 1 & 1 & 2 & 2 & 1 & 1 & 2 & 1 & 2 & 1 & 8 & 4 \\ 16 & 8 & 1 & 1 & 2 & 2 & 1 & 1 & 4 & 4 & 2 & 1 & 4 & 2 & 4 & 2 \\ 8 & 4 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 1 & 2 & 1 & 4 & 2 \end{bmatrix}.$$

One sees that every row of the matrix $\mathbf{M}$ contains at least five 1's, which means that every input block with a single non-zero byte will produce an output block with at least five non-zero bytes. The diffusion provided by the matrix $\mathbf{M}$ is extremely fast. Moreover, the matrix $\mathbf{M}$ is so designed as to make SAFER+ highly resistant to differential cryptanalysis, as will be explained in Section 5.2.

The PHT in the one-dimensional case has the matrix

$$\mathbf{H_1} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix},$$

which means that the first output byte is the sum modulo 256 of twice (or one left shift of) the first input byte and the second input byte, while the second output byte is just the sum modulo 256 of the two input bytes. Realization of this simple transformation corresponds to a "butterfly" in the usual language

of signal processing. The two-dimensional PHT has the matrix

$$\mathbf{H_2} = \begin{bmatrix} 4 & 2 & 2 & 1 \\ 2 & 2 & 1 & 1 \\ 2 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix},$$

which is just the Kronecker product of the matrix $\mathbf{H_1}$ with itself. This is realized by four PHT butterflys appropriately arranged in two levels. Similarly, the three-dimensional PHT and four-dimensional PHT are the threefold and fourfold Kronecker product of the matrix $\mathbf{H_1}$ with itself and can be realized with 16 and 64 PHT butterflys appropriately arranged in three levels and four levels, respectively. The matrix $\mathbf{M}$ was also chosen so that it can be realized by 64 PHT butterflys arranged in four levels, which is very desirable for efficient implementation whether in software or hardware, but the interconnection pattern of the signals between levels was chosen so as to achieve the above-described desirable properties of the resulting matrix.

Scalability

SAFER+ has the property, as do all ciphers in the previous SAFER family, that it can be scaled down to "mini-versions" that permit study of the properties of the algorithm in a simplified setting. A byte in standard SAFER+ can be reduced to 2 bits (and hence the block size reduced to 32 bits) by changing the modulo 256 and modulo 257 arithmetic used in SAFER+ to modulo 4 and modulo 5 arithmetic, respectively. Alternatively, a byte in standard SAFER+ can be reduced to 4 bits (and hence the block size reduced to 64 bits) by changing the modulo 256 and modulo 257 arithmetic used in SAFER+ to modulo 16 and modulo 17 arithmetic, respectively. [It is even possible to make a "maxi-version" of SAFER+ in which a byte in standard SAFER+ is expanded to 16 bits (and hence the block size increased to 256 bits) by changing the modulo 256 and modulo 257 arithmetic used in SAFER+ to modulo 2^{16} and modulo $2^{16} + 1$ arithmetic, respectively. This would in fact be a very strong cipher but rather difficult to implement since the logarithm and exponential tables would each require 2^{16} entries of 16 bits each.] The only "arbitrary" parameter that must be selected for these non-standard versions of SAFER+ is the element of multiplicative order 2^m, modulo the prime number $2^m + 1$ where m is the number of bits in a symbol, used as the base for exponentiation and logarithms. The choice of this parameter (which was chosen as 45 in standard SAFER+ where $m = 8$) is

of minor importance–any element of multiplicative order 2^m can be used in the m-bits/symbol mini-version of SAFER+.

The value of scalability is that it permits quick checking on mini-versions of conjectures about properties of the full SAFER+ algorithm. Its usefulness is well illustrated by the fact that a supposed and plausible weakness that had been asserted[1] to exist in SAFER K-64 could be demonstrated as erroneous by exhaustive testing of a mini-version of this cipher.

Biases in Key Schedule

The addition of "biases" to shifted versions of the user–selected key in order to generate round subkeys, which was described in Section 2.1.3, is used in SAFER+ to avoid "weak keys" such as can occur with the DEA of DES or with IDEA [6] where, for instance, if the user–selected key is all-zero then all round subkeys are identical. This property is of minor significance for encyption because of the negligibly small probability of choosing a "weak key", but it can be significant when the algorithm is used for hashing. Essentially, the addition of the biases in the SAFER+ key schedule randomizes the round subkeys generated from a single user–selected key and hence avoids the occurrence of such "weak keys".

Parity Word and Selections in Key Schedule

In SAFER K-64, the first cipher in the SAFER family [1], the key schedule differed from those of SAFER + as shown in Figs. 4–6 in that the same bytes of the key register were always selected for addition to the biases to produce the round subkeys. Knudsen [8] showed that this "stationarity" of the bytes in the key register made it possible, when SAFER K-64 was used in a hashing mode, to create "collisions" much more often than by random guessing. Knudsen suggested a modification of the key schedule to remove this weakness, which Massey then incorporated into the later ciphers (SAFER SK-64, SAFER SK-128, and SAFER SK-40) in the SAFER family after obtaining from Knudsen a signed statement relinquishing all intellectual rights to this modified key schedule. Knudsen's key schedule not only introduced the "non-stationarity" of the bytes of the key register added to the bias, which is implemented by the "selection" boxes in Figs. 4–6, but it also included expansion of the key register by the addition of a "parity byte," i.e., a byte which is the modulo-two sum of all the bytes in the user–selected key. This

[1] S. Murphy, "An Analysis of SAFER," preliminary manuscript dated June 6, 1995.

augmentation of the key register has the very desirable consequence that, if two user–selected keys differ, then they will differ in at least two bytes of the key register where the parity byte is also present. This provides a healthy diversity of the keys produced by the key schedule of SAFER+.

Number of Rounds

The most effective attack against SAFER+ appears to be Biham and Shamir's differential cryptanalysis [7]. As will be explained in Section 5.2, SAFER+ with six rounds is essentially immune to this attack. The choice of 8 rounds when a 128-bit key is used was made to provide a significant margin of safety for the security of SAFER+. This choice also has the desirable effect, as can be seen from Fig. 4, that each byte of the 16-byte user–selected key directly affects some round subkey exactly once in each of the 16 possible byte positions, as does also the parity byte appended to the user–selected bytes. The choice of 12 rounds for SAFER+ with a 192-bit key was made to provide the significant additional security to which a user is entitled who opts to use this longer key length. This choice also has the desirable effect, as can be seen from Fig. 5, that each byte of the 24-byte user–selected key again directly affects some round subkey exactly once in each of the 16 possible byte positions, as does also the parity byte appended to the user–selected bytes. The choice of 16 rounds for SAFER+ with a 256-bit key was made to provide the superabundant security to which a user is entitled who opts to use this longest key length. This choice also has the desirable effect, as can be seen from Fig. 6, that each byte of the 32-byte user–selected key once again directly affects some round subkey exactly once in each of the 16 possible byte positions, as does also the parity byte appended to the user–selected bytes.

4 Computational Efficiency of SAFER+

4.1 ANSI C with 200 MHz Pentium Platform

The data below specify the computational speed for the optimized ANSI C implementation of SAFER+.
Platform: Pentium Pro Processor, 200 MHz, 64 MB RAM, running Windows 95. (1 sec = 1000 clocks.)

Encryption
These data were obtained by timing the encryption of 1,000,000 blocks on the above described Pentium platform.

- SAFER+ with 128 bit key (8 rounds)–10,425 clocks per encrypted block, i.e., about 10.4 microseconds per encrypted block or about 12.3 megabits per second of encrypted data.
- SAFER+ with 192 bit key (12 rounds)–15,492 clocks per encrypted block, i.e., about 15.5 microseconds per encrypted block or about 8.3 megabits per second of encrypted data.
- SAFER+ with 256 bit key (16 rounds)–20,560 clocks per encrypted block, i.e., about 20.5 microseconds per encrypted block or about 6.2 megabits per second of encrypted data.

Decryption
These data were obtained by timing the decryption of 1,000,000 blocks on the above described Pentium platform.

- SAFER+ with 128 bit key (8 rounds)–10,425 clocks per decrypted block, i.e., about 10.4 microseconds per decrypted block or about 12.3 megabits per second of decrypted data.
- SAFER+ with 192 bit key (12 rounds)–15,492 clocks per decrypted block, i.e., about 15.5 microseconds per decrypted block or about 8.3 megabits per second of decrypted data.
- SAFER+ with 256 bit key (16 rounds)–20,560 clocks per decrypted block, i.e., about 20.6 microseconds per decrypted block or about 6.2 megabits per second of decrypted data.

Setting up of Key
These data were obtained by timing 1,000,000 executions of the SAFER+ key schedule on the above described Pentium platform.

- SAFER+ with 128 bit key–15,342 clocks, i.e., about 15.3 microseconds to run the key schedule.
- SAFER+ with 192 bit key (12 rounds)–28,622 clocks, i.e., about 28.6 microseconds to run the key schedule.
- SAFER+ with 256 bit key (16 rounds)–45,725 clocks, i.e., about 45.7 microseconds to run the key schedule.

Setting up of Algorithm
These data were obtained by timing 1,000,000 settings up of the three tables (exponentiation of 45 modulo 257, logarithms to the base 45, and key biases) required in the SAFER+ algorithm on the above described Pentium platform. The time is independent of the key length because, even for the shorter keys where not all entries in the bias table are needed, the entire bias table was generated each time.

- SAFER+ with 128 bit key–88,077 clocks, i.e., about 88 microseconds to set up the algorithm.
- SAFER+ with 192 bit key (12 rounds)–88,077 clocks, i.e., about 88 microseconds to set up the algorithm.
- SAFER+ with 256 bit key (16 rounds)–88,077 clocks, i.e., about 88 microseconds to set up the algorithm.

Changing Key after Initial Setup
These data are the same as those given above for setting up of a key.

4.2 8-bit Processors

The following SAFER+ computational efficiency estimates encryption, decryption, and setting up of a key for 8-bit processors are based on the MCS 51 family microcontrollers' Programmers Guide. A more detailed explanation of these data is given in the Appendix "SAFER+ Computational Efficiency Estimates for 8-bit Processors", which is attached to this document.

Encryption
The number of clocks required for encryption is estimated to be $r \times 10080$, where r is the number of rounds (8, 12 or 16).

- SAFER+ with 128 bit key (8 rounds)–80,640 clocks per encrypted block, i.e., for a 16 MHz clock frquency, about 5 milliseconds per encrypted block or about 25.6 kilobits per second of encrypted data.

- SAFER+ with 192 bit key (12 rounds)–120,960 clocks per encrypted block, i.e., for a 16 MHz clock frquency, about 7.6 milliseconds per encrypted block or about 16.9 kilobits per second of encrypted data.

- SAFER+ with 256 bit key (16 rounds)–161,280 clocks per encrypted block, i.e., for a 16 MHz clock frquency, about 10.1 milliseconds per encrypted block or about 12.7 kilobits per second of encrypted data.

Decryption
The number of clocks required for decryption is the same as for encryption.

- SAFER+ with 128 bit key (8 rounds)–80,640 clocks per decrypted block, i.e., for a 16 MHz clock frquency, about 5 milliseconds per decrypted block or about 25.6 kilobits per second of decrypted data.

- SAFER+ with 192 bit key (12 rounds)–120,960 clocks per decrypted block, i.e., for a 16 MHz clock frquency, about 7.6 milliseconds per decrypted block or about 16.9 kilobits per second of decrypted data.

- SAFER+ with 256 bit key (16 rounds)–161,280 clocks per decrypted block, i.e., for a 16 MHz clock frquency, about 10.1 milliseconds per decrypted block or about 12.7 kilobits per second of decrypted data.

Setting up of Key
The number of clocks required for entering a used-selected key and performing the key schedule algorithm is estimated to be $L \times (48 \times L + 4200)$, where L is the key length in bytes (8, 12 or 16).

- SAFER+ with 128 bit key (8 rounds)–36,672 clocks, i.e., for a 16 MHz clock frquency, about 2.3 milliseconds to run the key schedule.

- SAFER+ with 192 bit key (12 rounds)–57,312 clocks, i.e., for a 16 MHz clock frquency, about 3.6 milliseconds to run the key schedule.

- SAFER+ with 256 bit key (16 rounds)–79,488 clocks, i.e., for a 16 MHz clock frquency, about 5.0 milliseconds to run the key schedule.

Setting up of Algorithm
The following are conservative rough estimates for setting up of the three tables (exponentiation of 45 modulo 257, logarithms to the base 45, and key biases) required in the SAFER+ algorithm. The estimated time given is independent of the key length because, even for the shorter keys where not all entries in the bias table are needed, it would generally be convenient to generate the entire bias table each time.

- SAFER+ with 128 bit key (8 rounds)–120,000 clocks, i.e., for a 16 MHz clock frquency, about 7.5 milliseconds to set up the algorithm.
- SAFER+ with 192 bit key (12 rounds)–120,000 clocks, i.e., for a 16 MHz clock frquency, about 7.5 milliseconds to set up the algorithm.
- SAFER+ with 256 bit key (16 rounds)–120,000 clocks, i.e., for a 16 MHz clock frquency, about 7.5 milliseconds to set up the algorithm.

Changing Key after Initial Setup
These data are the same as those given above for setting up of a key.

4.3 Hardware Simulation

For simulated hardware implementation, SAFER+ has been implemented in VERILOG HDL by using Synplify tools.
The measurement tools and conditions used were:

- Synplify and MAX+Plus II;
- ALTERA chip with speed grade:-3 (80 MHz);
- System clock: 62 MHz.

The simulation results were as follows:

1. Number of Synopsys cells:
    - 62,000
2. Encryption and decryption rate for 128-bit key SAFER+:
    - 58.9 megabits per second.

5 Cryptographic Strength and Analysis

5.1 Cryptographic Strength of SAFER+

We are not aware of any cryptographic weaknesses in SAFER+. In our opinion, for all three key lengths (128 bits, 192 bits, and 256 bits) accomodated by the algorithm, there is no chosen-plaintext attack on SAFER+ more effective than exhaustive key search. Some of the principal reasons for this belief are set forth in the following sections.

5.2 Strength against Differential Cryptanalysis

Differential cryptanalysis is a general attack against an iterative cipher that was introduced by Biham and Shamir in 1990 [7]. Differential cryptanalysis has proved to be the most effective general attack against the previous SAFER family of ciphers, cf. [4], [8] and [9], and appears also to be the most effective general attack against SAFER+. In the below discussion of differential cryptanalysis, we follow the approach and notation of [4] and [6].

5.2.1 Differences and Differentials

As can be seen from Fig.2, at the beginning of a round, SAFER+ combines the 16-byte round input $\mathbf{X} = [X_1, X_2, \ldots, X_{16}]$ byte-wise with the 16-byte vector $\mathbf{Z}_a = [Z_{a1}, Z_{a2}, \ldots, Z_{a16}]$, which is first of the two round subkeys. This produces the 16-byte input $\mathbf{T} = [T_1, T_2, \ldots, T_{16}]$ to the nonlinear layer in the manner that $\mathbf{T} = \mathbf{X} \otimes \mathbf{Z}_a$ where "$\otimes$" is the product–group addition operator

$$\otimes = [\oplus, +, +, \oplus, \oplus, +, +, \oplus, \oplus, +, +, \oplus, \oplus, +, +, \oplus\ ]$$

in which $\oplus$ denotes bit-wise modulo-two addition of bytes and $+$ denotes usual byte addition, i.e., addition modulo 256.

Let the 16-byte vector $\mathbf{S} = [S_1, S_2, \ldots, S_{16}]$ denote the input to the linear layer described by the matrix $\mathbf{M}$ in Fig. 2. Note that S_j is given by

$$S_j = 45^{(X_j \oplus Z_{aj})} + Z_{bj}, \quad j \in \{1, 4, 5, 8, 9, 12, 13, 16\}$$

and

$$S_j = \log_{45}(X_j + Z_{aj}) \oplus\ Z_{bj}, \quad j \in \{2, 3, 6, 7, 10, 11, 14, 15\}$$

where $\mathbf{Z}_b = [Z_{b1}, Z_{b2}, \ldots, Z_{b16}]$ is the second of the two round subkeys. The output $\mathbf{Y} = [Y_1, Y_2, \ldots, Y_{16}]$ of the linear layer, which is also the round output, is given by $\mathbf{Y} = \mathbf{S}\,\mathbf{M}$.

Let "$\oslash$" denote the product–group subtraction of 16-byte vectors, i.e.,

$$\oslash = [\oplus, -, -, \oplus, \oplus, -, -, \oplus, \oplus, -, -, \oplus, \oplus, -, -, \oplus\ ]$$

where "$-$" denotes usual byte subtraction, i.e., subtraction modulo 256. Then the *difference* $\Delta\mathbf{V} = [\Delta V_1, \Delta V_2, \ldots, \Delta V_{16}]$ between the 16-byte words $\mathbf{V}$ and $\mathbf{V}^*$ is the 16-byte vector

$$\Delta\mathbf{V} = \mathbf{V} \oslash \mathbf{V}^*.$$

Equivalently,

$$\Delta\mathbf{V} = \begin{array}{cccc} [V_1 \oplus V_1^*, & V_2 - V_2^*, & V_3 - V_3^*, & V_4 \oplus V_4^*, \\ V_5 \oplus V_5^*, & V_6 - V_6^*, & V_7 - V_7^*, & V_8 \oplus V_8^*, \\ V_9 \oplus V_9, & V_{10} - V_{10}^*, & V_{11} - V_{11}^*, & V_{12} \oplus V_{12}^*, \\ V_{13} \oplus V_{13}^*, & V_{14} - V_{14}^*, & V_{15} - V_{15}^*, & V_{16} \oplus\ V_{16}^*]. \end{array}$$

Let $\mathbf{X}(i)$ and $\mathbf{Y}(i)$ denote the 16-byte input and output, respectively, of the i-th round of SAFER+. As usual in differential cryptanalysis, we assume that all round subkeys are chosen independently and uniformly at random (rather than actually produced by the SAFER+ key schedule from a random user–selected key). A pair (α, β) of *non-zero* 16-byte vectors, considered as the value of $(\Delta\mathbf{X}(1), \Delta\mathbf{Y}(i))$ is an *i-round differential* for SAFER+. It follows from Proposition 1 in [4] that SAFER+ is a *Markov cipher* as defined in [6], i.e., that the conditional probability

$$P(\Delta\mathbf{Y}(i) = \beta \mid \Delta\mathbf{X}(1) = \alpha,\ \mathbf{X}(1) = \gamma)$$

is independent of γ. This has the consequence that the sequence

$$\Delta\mathbf{X}(1),\ \Delta\mathbf{Y}(1),\ \Delta\mathbf{Y}(2), \ldots, \Delta\mathbf{Y}(i)$$

is a Markov chain and that the uniform distribution over non–zero differences is a stationary distribution for this Markov chain, cf. [6].

A sequence $(\alpha, \beta_1, \beta_2, \ldots, \beta_i)$, considered as the value of

$$(\Delta\mathbf{X}(1), \Delta\mathbf{Y}(1), \Delta\mathbf{Y}(2), \ldots, \Delta\mathbf{Y}(i))$$

is called an *i-round characteristic*. The probability

$$P(\Delta\mathbf{Y}(i) = \beta \mid \Delta\mathbf{X}(i) = \alpha)$$

of an i-round differential is just the sum of the probabilities

$$P(\Delta\mathbf{Y}(1) = \beta_1, \ldots, \Delta\mathbf{Y}(i-1) = \beta_{i-1}, \mathbf{Y}(i) = \beta \mid \Delta\mathbf{X}(1) = \alpha)$$

of the i-round characteristics $(\alpha, \beta_1, \ldots, \beta_{i-1}, \beta)$ for all non-zero values of β_1, β_2, ..., β_{i-1}. We will refer to these characteristics $(\alpha, \beta_1, \ldots, \beta_{i-1}, \beta)$ as *belonging to the i-round differential* (α, β).

5.2.2 Some Details of the Differential Cryptanalysis of SAFER+

The attack by differential cryptanalysis on an r-round cipher relies on being able to find an $(r-1)$-round differential whose probability is substantially greater than the average probability of such a differential, which is $\frac{1}{2^{128}-1} \approx 2^{-128}$ for a 16-byte block length. The existence of such a differential is generally characterized by the existence of a characteristic belonging to it that itself has a probability greater than this average differential probability. The task of showing the security of r-round SAFER+ against differential cryptanalysis is essentially that of showing that there are no $(r-1)$-round characteristics with probability greater than 2^{-128}. An exhaustive study of SAFER+ has shown that all 5-round characteristics have probability significantly smaller than 2^{-128} (but that this is not the case for only 4 rounds). The conclusion is that SAFER+ with six or more rounds (but not fewer) is secure against differential cryptanalysis.

Indeed, the matrix $\mathbf{M}$ of the linear layer, which was chosen to ensure that SAFER+ enjoys good diffusion (i.e., to ensure that small changes in round inputs cause large changes in round outputs) also ensures that "differences" similarly propagate and is the main source of the strength of SAFER+ against differential cryptanalysis. Let $\mathrm{W}(\mathbf{V})$ denote the number of non-zero bytes in the vector $\mathbf{V}$, which we will call the *weight* of $\mathbf{V}$. Because the operation of $\mathbf{M}$ is linear modulo 256 and because "differences" can be taken conveniently as byte differences modulo 256 at the output of the nonlinear layer in Fig. 2, diffusion is well measured by how well the PHT converts low weight inputs into high weight outputs.

To achieve the desired diffusion, the matrix $\mathbf{M}$ was chosen to have the following properties:

1. Each odd-numbered row of the matrix $\mathbf{M}$ is a permutation of the vector [16 8 4 4 4 4 2 2 2 2 2 1 1 1 1 1], and each even-numbered row is a permutation of the vector [8 4 4 2 2 2 2 2 1 1 1 1 1 1 1 1]. This fact results in two properties that strengthen the cipher against differential cryptanalysis:

   - Since the values 16, 8, 4, 2 and 1 are distributed almost "uniformly" over all columns of $\mathbf{M}$, good diffusion takes place already during one round.
   - If the input to the matrix $\mathbf{M}$ is a vector $\mathbf{S}$ with weight $\mathrm{W}(\mathbf{S}) = 1$ whose single non-zero byte has the value 128, then the output $\mathbf{Y} = \mathbf{S}\,\mathbf{M}$ will be a vector of weight either 5 or 8. This is important because byte differences of 128 tend to have relatively high probability.

2. A non-zero vector $\mathbf{V} = [V_1, V_2, \ldots, V_{16}]$, all of whose non-zero bytes occur in position pairs $(2j-1, j)$, $j \in 1, 2, \ldots 8$, will be called a *type*-A vector. If $\mathbf{V} = [V_1, V_2, \ldots, V_{16}]$ is a type-A vector and its non-zero bytes are all in its left half or all in its right half, then $\mathbf{V}$ will be called a *type*-B vector. If $\mathbf{V} = [V_1, V_2, \ldots, V_{16}]$ is a type-A vector and has non-zero bytes in both its left half and in its right half, then $\mathbf{V}$ will be called a *type*-C vector. The matrix $\mathbf{M}$ has the property that if $\mathbf{S}$ is a type-B vector with weight 2 or 4, then the output $\mathbf{Y} = \mathbf{S}\,\mathbf{M}$ will be a type-C vector with $\mathrm{W}(\mathbf{Y}) \geq 4$.

Both of these properties were exploited extensively in the above–mentioned exhaustive study of 5-round characteristics for SAFER+.

5.3 Strength against Linear Cryptanalysis

Linear cryptanalysis is a general attack against an iterative cipher that was introduced by Matsui in 1993 [10]. Linear cryptanalysis has proved to be a very effective general attack against ciphers in which the round subkeys are inserted by modulo-two addition. However, linear cryptanalysis is in general a weak attack against ciphers in which the round subkeys are inserted by addition modulo a larger modulus [12]. In particular, linear cryptanalysis is weak against the previous SAFER family of ciphers. With only three rounds, cf. pp. 41-47 in [11], these ciphers have been shown to be secure against linear cryptanalysis, cf. also [12] and [13].

Because SAFER+ inherits the structural properties that make the previous SAFER family of ciphers strongly secure against linear cryptanalysis, it is to be expected that SAFER+ likewise enjoys such security. In fact, SAFER+ is even stronger against linear cryptanalysis than are the ciphers in the previous SAFER family. With only two-and-one-half rounds, i.e., up to the input of the invertible linear transformation in the third encryption round, SAFER+ is already secure against linear cryptanalysis for the reasons that we now sketch.

5.3.1 Some Details of the Linear Cryptanalysis of SAFER+

We follow here the terminology and notation in [11] and [12].

A binary-valued function is *balanced* if it takes on the value 0 for exactly half of its possible arguments and the value 1 otherwise. An *I/O sum* $S^{(i)}$ for the i-th round of an iterative cipher is a modulo-two sum of a balanced binary-valued function f_i of the round input $X^{(i)} = Y^{(i-1)}$ and a balanced binary-valued function g_i of the round output $Y^{(i)}$, i.e.,

$$S^{(i)} = f_i(Y^{(i-1)}) \oplus g_i(Y^{(i)})$$

where $\oplus$ denotes modulo-two addition, i.e., the XOR operation. I/O sums for successive rounds are said to be *linked* if the output function of each I/O sum, except the last, coincides with the input function of the following I/O sum (i.e., $g_i = f_{i+1}$). When $S^{(1)}$, $S^{(2)}$,..., $S^{(\rho)}$ are linked, then their sum is also an I/O sum, namely

$$S^{(1\ldots\rho)} = \bigoplus_{i=1}^{\rho} S^{(i)} = g_0(Y^0) \oplus g_\rho(Y^{(\rho)}),$$

which is called a ρ-round I/O sum .

A homomorphism from a group $(B^n, \oplus)$ onto the group $(B, \oplus)$ is called a *binary-valued homomorphism* for $\otimes$. The binary-valued function f is such a homomorphism if f is not identically zero and if, for all U and V in B^n, $f(U \otimes V) = f(U) \oplus f(V)$. An I/O sum for rounds i to $j (j \geq i)$ is *homomorphic* if the input function is a binary-valued homomorphism for $\otimes_i$ and the output function is a binary-valued homomorphism for $\otimes_{j+1}$.

The *imbalance* $I(V)$ of binary random variable V is the non-negative real number $|2P[V = 0] - 1|$ where $P[V = 0]$ is the probability that V takes on the value 0. The success of the attack by linear cryptanalysis on

an r-round iterative cipher depends on being able to find an $r-1$ round homomorphic I/O sum with substantial imbalance. We have been able to prove that Harpes' procedure [11] for finding effective homomorphic I/O sums, which is the only practical procedure known, cannot find an I/O sum with non-zero imbalance for one-and-one-half rounds of SAFER+. Based on our experience, we believe that there is no homomorphic I/O sum whatsoever with non-negligible imbalance for one-and-one-half rounds of SAFER+, i.e., SAFER+ is already secure again linear cryptanalysis after only two-and-one-half rounds.

5.4 Other Analyses of SAFER+

5.4.1 Weak Keys

It is virtually certain that there are no "weak keys" in SAFER+ for any reasonable definition of this somewhat elusive cryptanalytic concept. The reason for this is the use of biases as described in Section 2.1.3 to "randomize" the round subkeys produced from the user-selected key. For this reason, there are no pairs of keys (such as binary complements) that have similar encrypting or decrypting properties. The result is that no precautions must be taken when selecting the input key for SAFER+; the best method is to choose the key uniformly at random, i.e., by coin-tossing.

5.4.2 Trap Doors

There are no "trap doors" built into SAFER+ that would allow one privy to knowledge of such a trap door to read encrypted traffic illicitly. The best "proof" of this is given in the remark labelled "Transparency" in Section 6 below.

5.5 Previous Cryptanalyses

Although SAFER+ is a new cipher and has not yet been subject to cryptanalysis except by its designers, there has been several cryptanalyses of the previous SAFER family of ciphers. These are given in [4], [5], [8], [9], [11], [12] and [13]. These cryptanalyses are all referred to within this document.

6 Advantages and Limitations of SAFER+

The major advantages that would be offered by the choice of SAFER+ for the Advanced Encryption Standard (AES) are:

- **A proven track record of security**–Many cryptanalysts in the public arena have attacked the previous ciphers in the SAFER family since their announcement in 1993. The only weakness found to date was in the original key schedule, as described above in Section 3, which led to a significantly improved key schedule that has been incorporated into SAFER+. As a result of cryptanalyses that have already been carried out, we may safely conclude that SAFER+ is secure against both differential cryptanalysis and linear cryptanalysis (in the sense that these attacks are not stronger than exhaustive key search), the two most powerful general attacks on iterative block ciphers known today.

- **Speed and simplicity**–The simple structure of SAFER+, which is based on byte operations only, makes it easy to implement correctly even by inexperienced programmers. It also makes possible a host of "tricks" that experienced programmers can use to speed up encryption and decryption. We expect that the "optimized" ANSI C implementations of SAFER+ included with this report will be significantly accelerated as other programmers concern themselves with the implementation of SAFER+.

- **Transparency**–The fact that the SAFER+ algorithm uses only well-defined mathematical functions (rather than "random-looking" tables) should convince any fair-minded person that there are no "trapdoors" (i.e., hidden weaknesses) built into this cipher for the benefit of those "in the know." There are simply no suspicious steps anywhere in the algorithm and every step has a clear cryptographic justification.

- **Flexibility of Use**–SAFER+ offers the same flexibility for usage as does the Data Encryption Algorithm (DEA) of the Data Encryption Standard (DES) [2]. In particular, it can be used in Electronic Code-Book (ECB) mode, in Cipher Block Chaining (CBC) mode, in Cipher FeedBack (CFB) mode and in Output FeedBack (OFB) mode in exactly the same manner as the DEA. It can also be used in hashing in accordance with the schemes described in [14]. SAFER+ can also be

used to generate a Message Authentication Code (MAC) and to produce pseudo-random numbers in essentially the same manner that the DEA can be used for these purposes.

It has not yet been done, but it would be easy to accomodate other key lengths into SAFER+ besides those adopted in this document (16 bytes, 24 bytes and 32 bytes) should this be desirable for some reason. However, it is not possible to change the block length from its present 16 bytes without a complete redesign of the algorithm.

- **Flexibility of Environment**–SAFER+ was designed to implement easily on any platform from an 8-bit processor upwards. The computational efficiency data given in Section 4 are evidence of this flexibility. The fact that only byte operations are performed within the algorithm and that its memory requirements are small makes it attractive for use on smart cards with 8-bit processors, in ATM, for HDTV, in B-ISDN, in voice applications, aboard satellites, and in virtually any other environment where encryption might be desirable.

The limitations of SAFER+ are:

- **No proof of complete security**–As for every cipher today in which the key is used more than once, there is no *proof* that SAFER+ is not vulnerable to some attack of a kind not as yet known.

- **Encryption/Decryption Dissimilarity**–The fact that SAFER+ is a general substitution/linear-transformation cipher, as was explained in Section 3 above, implies that its decrypting algorithm differs from the encrypting algorithm by more than just a change in the key schedule. However, it should be clear from Sections 2.1.1 and 2.1.2, as well as from Figs. 2 and 3, that there are substantial similarities between the encrypting and decrypting algorithms of SAFER+ so that one need pay only a small price in software or hardware for implementing both. Moreover, as was discussed in Section 3 above, this dissimilarity of encryption and decryption in SAFER+ was a source of desirable additional freedom in designing this cipher.

7 Intellectual Property Rights

SAFER+ is a completely non-proprietary algorithm. Neither Cylink Corporation nor any of the inventors have retained any intellectual property rights. The C and Java programs included in this report and the submitted diskettes are not copyrighted.

8 Diskettes Submitted with This Proposal

- Diskette #1 of 6—PDF file of this document.
- Diskette #2 of 6—ANSI C Reference Code for SAFER+.
- Diskette #3 of 6—Optimized ANSI C Implementation of SAFER+.
- Diskette #4 of 6—Optimized Java Implementation of SAFER+.
- Diskette #5 of 6—Test Values for SAFER+.
- Diskette #6 of 6—HDL Implementation of SAFER+.

References

[1] J. L. Massey, "SAFER K-64: A Byte-Oriented Block-Ciphering Algorithm," pp. 1-17 in *Fast Software Encryption* (Ed. R. Anderson), Lecture Notes in Computer Science No. 809. New York: Springer, 1994.

[2] U.S. Department of Commerce/National Bureau of Standards, FIPS Pub 46, Data Encryption Standard, April 1977.

[3] C.E. Shannon, "Communication Theory of Secrecy Systems", *Bell System Tech. J.*, vol. 28, pp. 656-715, Oct., 1949.

[4] J. L. Massey, "SAFER K-64: One Year Later," pp. 212-241 in *Fast Software Encryption II* (Ed. B. Preneel), Lecture Notes in Computer Science No. 1008. New York, Springer 1995.

[5] S, Vaudenay, "On the Need for Multipermutations Cryptanalysis of MD4 and SAFER," pp. 286-297 in *Fast Software Encryption II* (Ed. B. Preneel), Lecture Notes in Computer Science No. 1008. New York, Springer 1995.

[6] X. Lai, J. L. Massey and S. Murphy, "Markov Ciphers and Differential Cryptanalysis," pp. 17-38 in *Advances in Cryptology - EUROCRYPT '91* (Ed. D. W. Davies), Lecture Notes in Computer Science No. 547. Heidelberg and New York: Springer, 1991.

[7] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," pp. 212-241 in *Advances in Cryptology - CRYPTO '90* (Eds. A. J. Menezes and S. A. Vanstone), Lecture Notes in Computer Science No. 537. Heidelberg and New York: Springer 1990.

[8] L. R. Knudsen, "A Key-Schedule Weakness in SAFER K-64," in *Advances in Cryptology - CRYPTO '95* (Ed. D. Coppersmith), Lecture Notes in Computer Science No. 963. Heidelberg and New York: Springer, 1995.

[9] L. R. Knudsen and T. A. Berson, "Truncated Differentials of SAFER," pp. 15-26 in *Fast Software Encryption* (Ed. D. Gollmann), Lecture Notes in Computer Science, No. 1039, Heidelberg and New York: Springer, 1996.

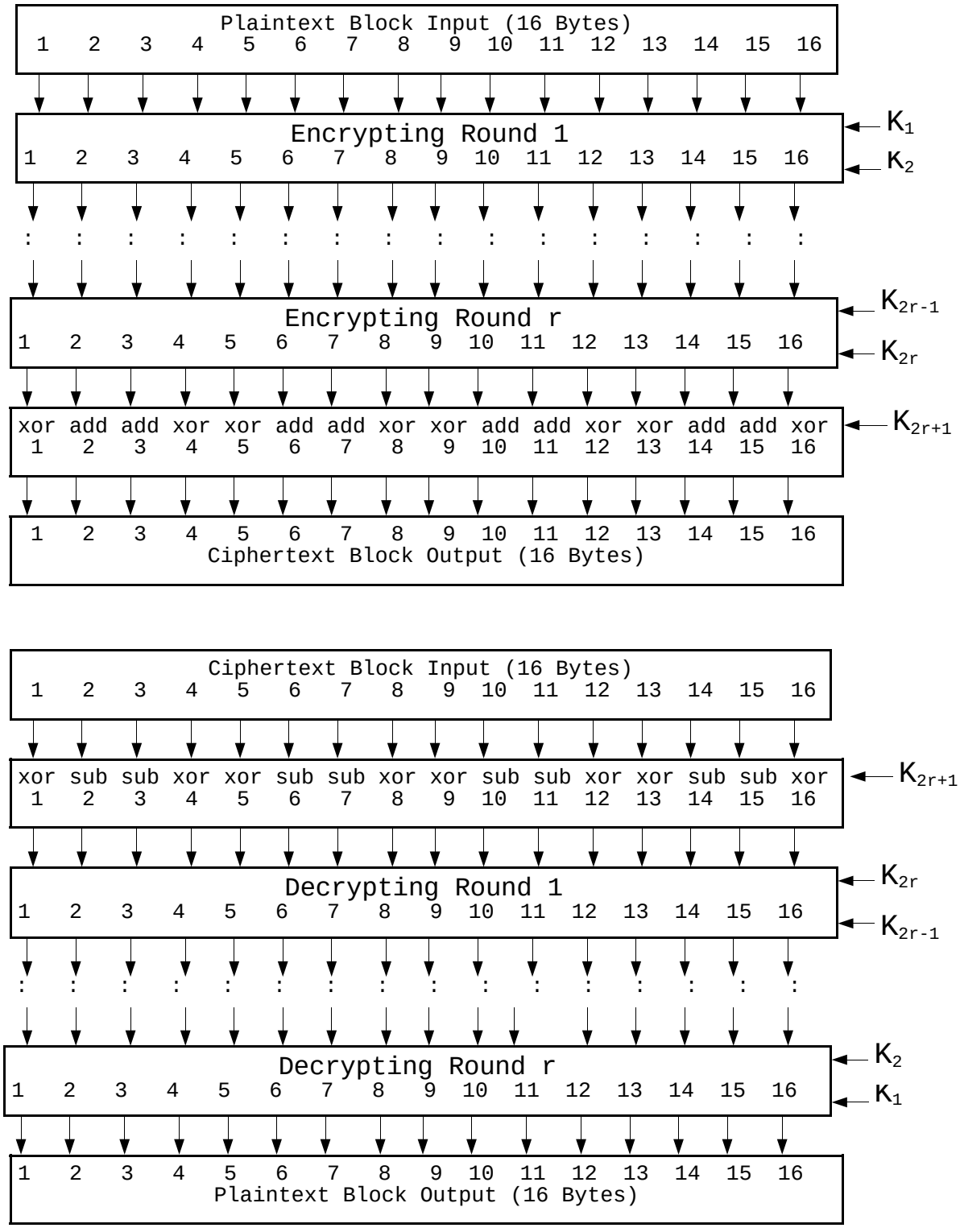
[10] M. Matsui, "Linear Cryptanalysis Method for DES Cipher," pp. 386-397 in *Advances in Cryptology-EUROCRYPT '93* (Ed. T. Helleseth), Lecture Notes in Computer Science No. 765. New York: Springer, 1994.

[11] C. Harpes, "Cryptanalysis of Iterated Block Ciphers," ETH Series in Inform. Processing. Konstanz: Hartung-Gorre Verlag, 1996.

[12] C. Harpes, G. C. Kramer and J. L. Massey, "A Generalization of Linear Cryptanalysis and the Applicability of Matsui's Piling-Up Lemma," pp. 24-38 in *Advances in Cryptology-EUROCRYPT'95* (Eds. L. C. Guillou and J.-J. Quisquater), Lecture Notes in Computer Science, No. 921. Heidelberg and New York: Springer, 1995.

[13] S. R. Perkins, "Linear Cryptanalysis of the SAFER K-64 Block Cipher," Semesterarbeit No. 9415, Institut für Signal- und Informationsverarbeitung, ETH Zürich, 15 July 1994.

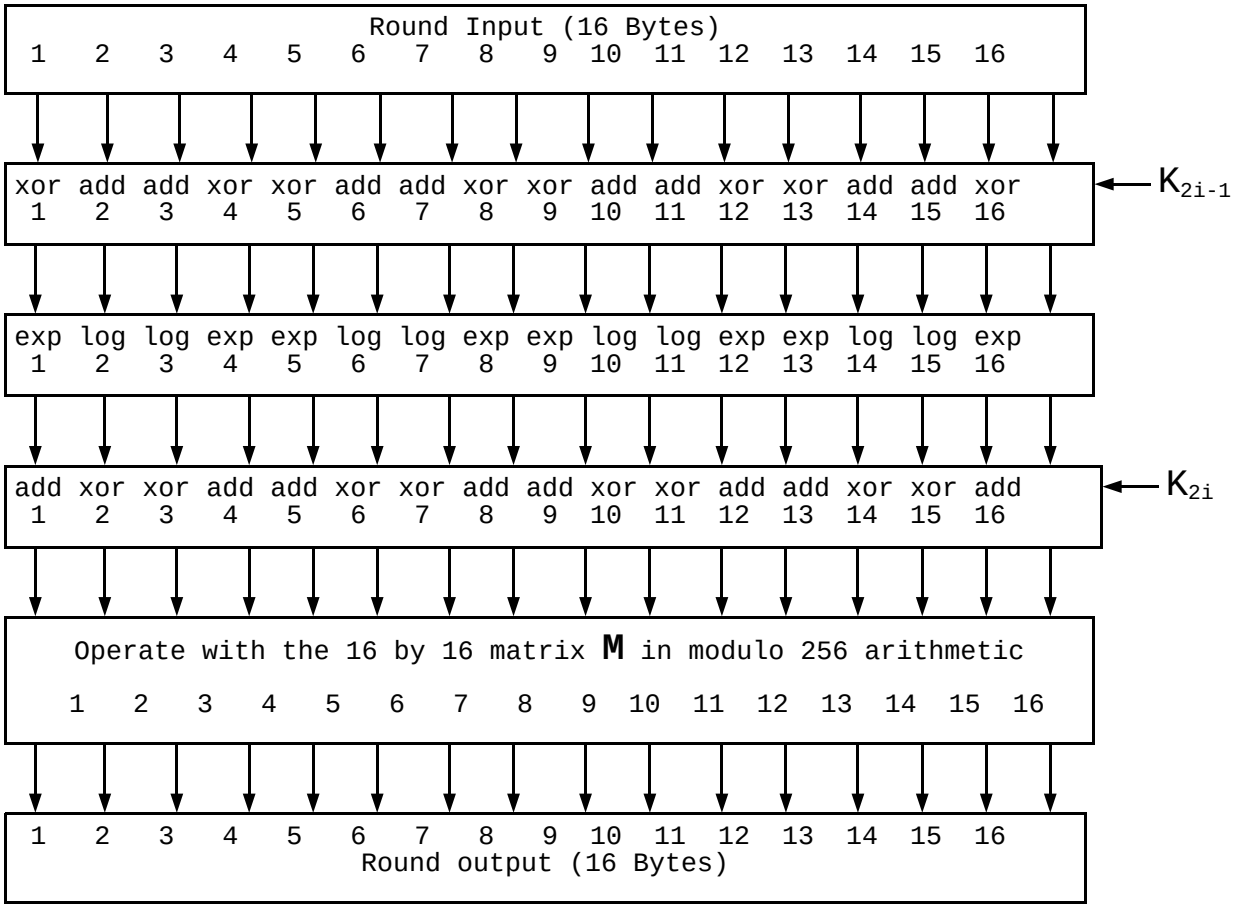
[14] X. Lai and J. L. Massey, "Hash Functions Based on Block Ciphers," pp. 55-70 in *Advances in Cryptology-EUROCRYPT'92* (Ed. R. A. Rueppel), Lecture Notes in Computer Science, No. 658. Heidelberg and New York: Springer, 1993.



Explanatory Notes

"xor" denotes bit-by-bit exclusive-or of bytes
"add" denotes modulo 256 addition of bytes
"sub" denotes modulo 256 subtraction of key byte from input byte

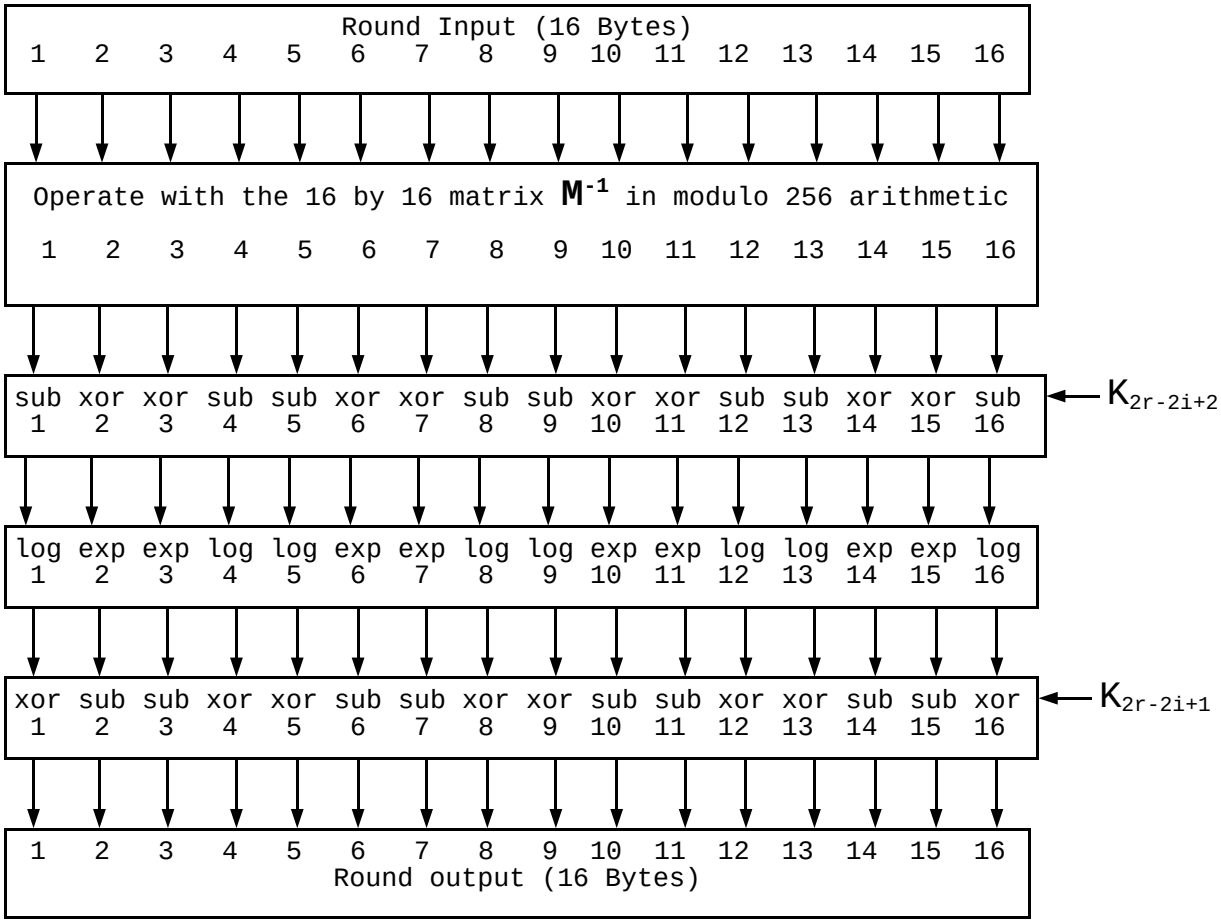
Fig. 1: Encrypting and Decrypting structure of the SAFER+ algorithm.



Explanatory Notes

"xor" denotes bit-by-bit exclusive-or of bytes
"add" denotes modulo 256 addition of bytes
"exp" denotes 45^x modulo 257, where x is the input byte, with the convention that the result $45^{128} = 256$ is represented by 0.
"log" denotes $\log_{45}(x)$, where x is the input byte, with the convention that $\log_{45}(0) = 128$.

Fig. 2: The structure of the SAFER+ algorithm for encryption round i.



Explanatory Notes

"xor" denotes bit-by-bit exclusive-or of bytes

"sub" denotes modulo 256 subtraction of key byte from input byte

"exp" denotes 45^x modulo 257, where x is the input byte, with the convention that the result $45^{128} = 256$ is represented by 0.

"log" denotes $\log_{45}(x)$, where x is the input byte, with the convention that $\log_{45}(0) = 128$.

Fig. 3: The structure of the SAFER+ algorithm for decryption round i.

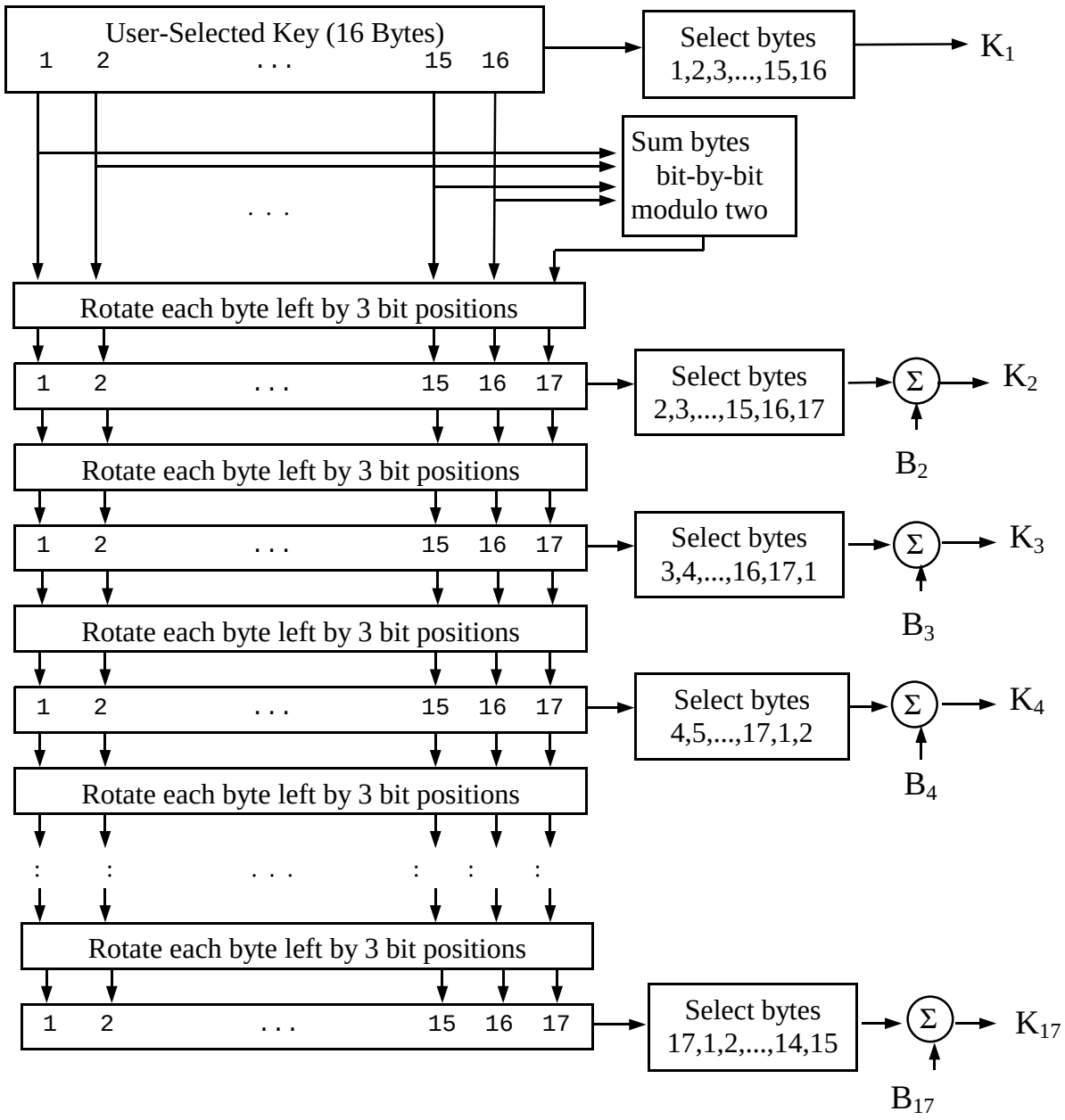
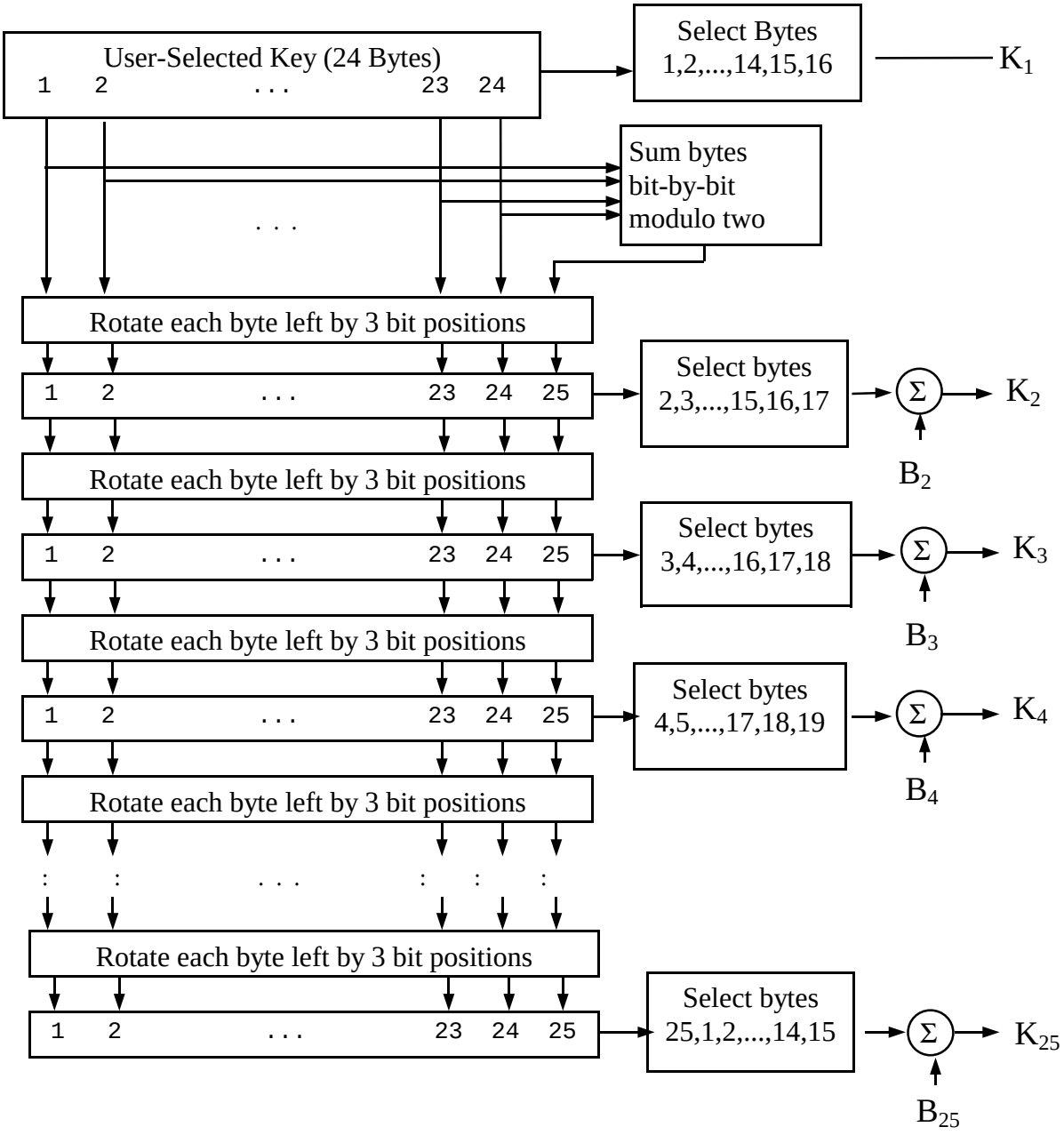
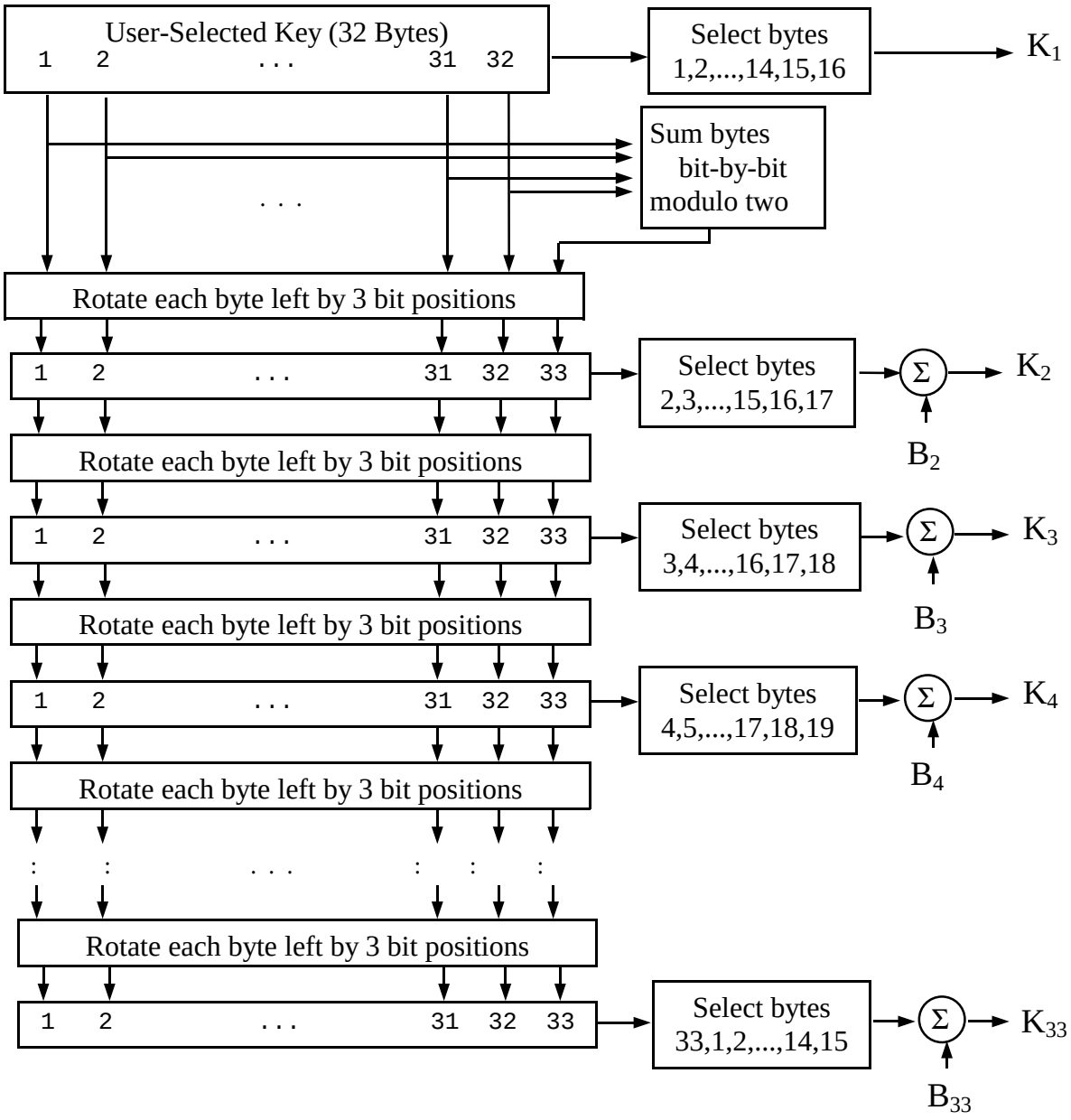


Fig. 4: SAFER+ key schedule for 128 bit key.



Σ denotes bytewise modulo 256 addition of 128 bit

Fig. 5: SAFER+ key schedule for 192 bit key.



Σ denotes bytewise modulo 256 addition of 128 bit

Fig. 6: SAFER+ key schedule for 256 bit key.

APPENDIX

SAFER+ Computational Efficiency Estimates for 8-bit Processors

For SAFER+, computational efficiency estimates based on the MCS 51 family microcontrollers' Programmers Guide have been used: http://developer.intel.com/design/MCS51/MANUALS/272383.htm

1. SAFER+ Encryption/Decryption timing estimates.

We suppose the following Internal memory organization during of ciphering .

	0	1	2	3	4	5	6	7
78h								
70h								
68h								
60h								
58h								
50h								
48h 40h	This part contains the two 16 byte keys for the current round, the first key starts at the Address 30h. It also contains any temporary variables .							
38h 30h								
28h	Exp. Table Address		Log. Table Address					
20h	Input Address		Output Address		Keys Table Address			
18h 10h 08h 00h	Stack SP initialized to 10h Bank0 and Bank1 used for current calculations (at first contains input)							

Each round of Encryption (or Decryption) is divided into 6 steps.

Step 1. During this step, the two 16 byte keys of the current round are read from External memory. This requires at least one MOVX (for getting one byte from External memory into the Accumulator), one MOV (for writing that byte from Accumulator to Internal memory), two INC operations (for the next byte address calculation) and a check of the cycle counter along with jumping. These operations are repeated 32 times. In addition, 120 **oscillator periods (o.p.)** are needed for the startup initialization of this step. As the result, the timing estimate for this step is 3192 o.p.

```
G1: MOVX   A , @DPTR      (24 o.p.)
    MOV    @Register , A  (12 o.p.)
    INC    Register       (12 o.p.)      ( 96 o.p. )
    INC    DPTR           (24 o.p.)
    DJNZ   Register , G1: (24 o.p.)
```

Step 2 and **Step 4** correspond to the ADD/XOR addition of the two round keys within SAFER+. Each (ADD,XOR,SUB) operation requires two MOV and one arithmetic operation (ADD,XOR,SUB). The timing estimate for this step is 1152 o.p. because there are 32 similar linear arithmetic operations.

```
    MOV   A , direct   (12 o.p.)
    OP    A , direct   (12 o.p.)    ( 36 o.p. )
    MOV   direct , A   (12 o.p.)
```

where OP is ADD,XRL or SUBB as appropriate.

Step 3. During this step, the EXP/LOG nonlinear transformations of SAFER+ are done.
One nonlinear transformation can be done in 108 o.p. The total number of such transformations is 16 .
The timing estimation for this step is 1728 o.p.

```
MOV   A , direct      ( 12 o.p.)
ADD   A , direct      ( 12 o.p.)
MOV   direct , A      ( 12 o.p.)
MOV   A , direct      ( 12 o.p.)
ADDC  A , direct      ( 12 o.p.)
MOV   direct , A      ( 12 o.p.)
MOVX  A , @DPTR       ( 24 o.p.)
MOV   direct , A      ( 12 o.p.)
```

(108 o.p.)

Step 5. This step corresponds to the linear transformation by the matrix **M** within the encryption round (or by the matrix $\mathbf{M}^{-1}$ within the decryption round) of SAFER+. This is implemented by "butterflys" having two input bytes and two output bytes. Each butterfly box requires 3 MOV and two ADD operations taking 60 o.p. There are 32 butterflys in all. Another 240 o.p. are needed for other required operations. The timing estimate for this step is 2160 o.p.

```
MOV   A , direct      ( 12 o.p.)
ADD   A , direct      ( 12 o.p.)
MOV   direct , A      ( 12 o.p.)
ADD   A , direct      ( 12 o.p.)
MOV   direct , A      ( 12 o.p.)
```

(60 o.p.)

Step 6. During this step the results of encryption (or decryption) are written to External memory. The timing is the same as for **Step 1** except that the number of moved bytes is now only 16. timing estimate for this step is thus 2040 o.p..

Thus, the timing estimate for one complete round of SAFER+ is 10080 o.p..
The total encryption (or decryption) time may be calculated as $T = r*10080*/(f*10^6)$ sec. where
f – timing clock frequency (12 MHz , 16 MHz)
r – number of rounds (8,12, or16)
For example, if f = 16 MHz and r = 8, then T = 0.005 sec (200 encryptions per second).

2. SAFER+ Key Schedule timing estimation

Initialization Step. During this step, the key material (user-defined key) is read from External memory. This step is the same as **Step1** of encryption except for the number of read bytes. Here, the number of read bytes is equal to the user-selected key length and can be 16, 24 or 32. The timing estimate for this step is thus keylength*96 o.p.

Each iteration within the key schedule is divided into 3 steps.
Step 1. This step correspond to the 3 bit left rotation part of key schedule .

```
MOV   A , direct      ( 12 o.p.)
RL    A               ( 12 o.p.)
RL    A               ( 12 o.p.)
```

(48 o.p.)

RL A (12 o.p.)

The timing estimate for this step is keylength*48 o.p.

Step 2 During this step, the result of previous step is added to the bias which is read from External memory .

```
G2:   MOVX   A , @DPTR    (24 o.p.)
      ADD    A , @R0      (12 o.p.)
      MOV    @R1 , A      (12 o.p.)
      INC    R0           (12 o.p.)
      INC    R1           (12 o.p.)
      INC    DPTR         (24 o.p.)
      JDNZ   R7 , G2:     (24 o.p.)
```

(120 o.p.)

These operations are repeated 16 times .

The timing estimate for this step is 1920 o.p.

Step 3 During this step, the current expanded key is written to External memory as in Step 6 of encryption. The timing estimate for this step is 2040 o.p.

Hence, the timing estimate for one iteration is keyLength*48 + 1920 + 2040 o.p..
Another 240 o.p. are added to account for other timing delays .

The timing estimate for the total key schedule is $T = keyLength*(keyLength*48 + 4200) /(f*10^6)$ sec where
f – timing clock frequency (12 MHz , 16 MHz)
keyLength – user-selected key length in bytes .
For example, if f = 16 MHz and keyLength = 16, then T = 0.004 sec .