SAFER K–64: One Year Later

James L. Massey

Signal & Information Processing Laboratory
Swiss Federal Institute of Technology
ETH Zentrum
CH-8092 Zurich, Switzerland

1 Introduction

Since we introduced the cipher SAFER K–64 (an acronym for **S**ecure **a**nd **F**ast **E**ncryption **R**outine with a user–selected **K**ey of **64** bits) one year ago at the predecessor to this workshop [MAS94], we have been pleasantly surprised by the rapidity of its acceptance within the cryptographic users' community. Undoubtedly the foremost reason for this is the non-proprietary character of SAFER K–64, which makes it unusally attractive to users. Although our design of SAFER K–64 was sponsored by Cylink Corporation (Sunnyvale, CA, USA), Cylink has explicitly relinquished any proprietary rights to this algorithm. This largesse on the part of Cylink was motivated by the reasoning that the company would gain more from new business than it would lose from competition should many new users adopt this publicly available cipher. SAFER K–64 has not been patented and, to the best of our knowledge, is free for use by anyone without fees of any kind and with no violation of any rights of ownership, intellectual or otherwise. Indeed, one way in which we have become aware of applications of SAFER K–64 is via the requests that we have received from users for written assurance of the non–proprietary character of SAFER K–64 (and of SAFER K–128 that is described in the next section).

Almost immediately upon the announcement of SAFER K–64, we began to receive requests for a version of this cipher with a 128–bit user–selected key. In many ways, 128 is a natural key length because the cipher uses 128 bits from the key schedule within each round. The Special Projects Team of the Ministry of Home Affairs, Singapore, took the initiative to design a key schedule to be used with the basic SAFER algorithm for a 128–bit user–selected key. We found their key schedule to be very attractive because, when the two halves of the 128–bit key are the same 64–bit string, it produces the same round keys as does the key schedule for SAFER K–64 when its user–selected key is this same 64–bit string. The designers have renounced all proprietary rights to this 128–bit key schedule and have authorized us both to announce their key schedule and to standardize its use. We do this in Section 2 of this paper where we refer to the resultant cipher as SAFER K–128. Hereafter, we will say simply 'SAFER' when our remarks apply to both SAFER K–64 and SAFER K–128.

A second factor in the quick popularity of SAFER is its byte orientation. Within the enciphering and deciphering processes, all operations are on bytes, which makes SAFER especially attractive for implementation on smart cards

with 8-bit internal processors. This fact played an important role in the tentative selection by Singaporean planners of SAFER K–128 as the standard cipher within the island–wide information system being planned for the turn of the century. A prototype smart-card implementation of SAFER was found there to run about 2.5 times as fast as a fully optimized smart-card implementation of the Data Encryption Standard (DES).

We have received several enquiries about our reasons for choosing the 'logarithm' and 'exponential' functions to provide the 'nonlinearities' in SAFER that are required for good 'confusion'. To answer these questions, we give in Section 3 an analysis to show that these functions well resemble 'randomly chosen' functions. Further justification for the choice of these nonlinearities is given in the paper [VAU95] in this volume, which shows that other choices would have given a much weaker cipher.

One of the novel features of SAFER was the use of a new linear transform to provide the "diffusion" that a good cipher requires, i.e., to ensure that small changes in each round input result in large changes in the round output. We called this transform the Pseudo-Hadamard Transform (PHT) as it differs from the conventional Hadamard (or Walsh-Hadamard) transform only enough to make it invertible over the ring of integers modulo 256. Again we have been questionned, sometimes skeptically, as to how good this diffusion is. In Section 4, we give a detailed discussion of the diffusing capability of the PHT, not only to answer these questions but also because the results are essential to the cryptanalysis in Section 6. We were remiss in [MAS94] in not mentionning two earlier applications in cryptography of transform techniques similar to the Hadamard transform and we are pleased to remedy this omission here. Huber [HUB90] also used the "butterfly with decimation" structure of the Hadamard transform within an encryption round to provide diffusion, but replaced the linear "butterflys" with two–input two–output nonlinear functions to obtain the required invertibility of the transform. Schnorr, in a paper presented in the rump session at CRYPTO '91, cf. [SCH92], used the "butterfly with decimation" structure of the fast Walsh-Hadamard transform to obtain diffusion within a hashing function.

For a cipher to gain popularity, there must be a general belief that it is 'secure'. The resistance of a cipher to differential cryptanalysis, introduced by Biham and Shamir [BIH90], is perhaps the best measure available today of its security.We are aware of several privately conducted and proprietary differential cryptanalyses of SAFER, all of which have reached the conclusion that SAFER is secure against differential cryptanalysis, but there has been some disagreement about how many rounds of SAFER are required for this security. In [MAS94], we recommended the use of six rounds in SAFER K–64 but allowed optionally up to ten rounds. In Section 6 we give our own detailed differential cryptanalysis, which shows that six rounds of SAFER K–64 suffices for protection against differential cryptanalysis. The next best measure today of a cipher's security is its resistance to linear cryptanalysis, introduced by Matsui [MAT93, ?]. We have had no reports from others on the strength of SAFER against linear cryptanaly-

sis, but together with our students [PER94, ?, ?] we have undertaken the linear cryptanalysis of SAFER. Because of the lengthy treatment that is required to do justice to the differential cryptanalysis of SAFER, we will not discuss this work further here, except to mention that it indicates that SAFER is even more secure against linear cryptanalysis than against differential cryptanalysis, which is the reverse of the situation for DES.

Very recently, Knudsen [KNU95] has pointed out a 'weakness' in SAFER when this cipher is used within a public hashing scheme. We discuss this 'weakness' in Section 7 where we also give a specification for its avoidance. We close in Section 8 with some remarks.

2 SAFER K–128

SAFER K–64 with r rounds uses $2r+1$ 64–bit subkeys that are derived from the 64–bit user–selected key according to the key schedule shown in Fig. 1. We now define SAFER K–128 as the cipher whose encryption round structure, output transformation and key biases are identical to those of SAFER K–64 but whose $2r+1$ 64–bit subkeys are derived from the 128–bit user–selected key according to the key schedule shown in Fig. 2. As mentioned above, this latter key schedule was designed by the Special Projects Team of the Ministry of Home Affairs, Singapore. We recommend that $r = 10$ rounds of encryption be used with SAFER K–128 and specify that not more than 12 rounds be used.

The left and right halves of the 128–bit user–selected key are denoted as $K_{\rm a}$ and $K_{\rm b}$, respectively, in Fig. 2 where, as in [MAS94], we abide by the convention that more significant bits and bytes are to the left. Upon comparing Figs. 1 and 2, one sees immediately that if the righthalf key $K_{\rm b}$ in Fig. 2 coincides with the 64–bit user–selected key K_1 in Fig. 1, then the same subkeys K_1, K_3, K_5, ... are generated by both key algorithms. Similarly, if the lefthalf key $K_{\rm a}$ in Fig. 2 coincides with the 64–bit user–selected key K_1 in Fig. 1, then the same subkeys K_2, K_4, K_6, ... are generated by both key algorithms. Thus, when *both* $K_{\rm a}$ and $K_{\rm b}$ in Fig. 2 coincide with the 64–bit user–selected key K_1 in Fig. 1, then *all* subkeys produced by both key schedules are the same. This is a very desirable feature as it permits a user with an implementation of SAFER K–128 to encipher and decipher for SAFER K–64 whenever desired.

Appendix B contains a TURBO PASCAL program that implements encryption for the full r-round SAFER K–128 cipher. This program should be taken as the official definition of the SAFER K–128 encryption algorithm. Appendix C gives two examples of 12–round encryption (i.e., $r = 12$) that the reader may find useful in checking his or her own implementation of this cipher.

Fig. 1. Key Schedule for SAFER K–64.

3 The Nonlinearities of SAFER

We begin by recalling the encryption round structure of SAFER shown in Fig. 3. The first step within the i-th round is the Mixed XOR/Byte-Addition of the round input with the subkey K_{2i-1}. The eight resulting bytes are then individually subjected to one of two different transformations, namely: (1) the operation labelled "$45^{(\cdot)}$" in Fig. 3 to denote that if the input byte is the integer j then the output byte is 45^j modulo 257 (except that this output is taken to be 0 if if the modular result is 256, which occurs for $j = 128$) and (2) the operation labelled "$\log_{45}$" in Fig. 3 to denote that if the byte is the integer j then the output byte is $\log_{45}(j)$ (except that this output is taken to be 128 if the input is $j = 0$), i.e., the power to which one must raise 45 to obtain j modulo 257. Because 257 is a prime, arithmetic modulo 257 is the arithmetic of the finite field GF(257). The element 45 is a primitive element of this field, i.e., its first 256 powers generate all 256 non–zero field elements. Thus the mapping $45^{(\cdot)}$ is an invertible mapping from one byte to one byte. The mapping $\log_{45}(.)$ is just the inverse of the mapping $45^{(\cdot)}$.

To see just how "nonlinear" these two mappings are or, better, how closely

Fig. 2. Key Schedule for SAFER K–128.

Fig. 3. Encryption round structure of SAFER.

they resemble a "randomly chosen" mapping, we consider for each mapping the boolean functions that determine each output bit in terms of the eight input bits. Any boolean function of 8 input bits, say $f(.)$, has an *algebraic normal form* (ANF) of the type

$$\begin{aligned} f(x_1, x_2, \ldots x_8) = a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_8 x_8 \\ + a_{1,2} x_1 x_2 + a_{1,3} x_1 x_3 + \cdots + a_{7,8} x_7 x_8 \\ + \cdots + a_{1,2,3,4,5,6,7,8} x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 \ . \end{aligned} \tag{1}$$

The coefficients on the right are elements of the finite field GF(2) and the addition is addition in this field, which is just the XOR operation. The *nonlinear order* of a product of variables is the number of variables in that product; the nonlinear order of the function itself is the maximum nonlinear order of a product of variables appearing with a non–zero coefficient in its ANF. Each boolean function of eight bits uniquely determines the coefficients of its ANF and, conversely, any choice of these coefficients determines such a function. Choosing such a function f uniformly at random from the set of all 2^{256} such functions is thus equivalent to choosing the coefficients on the right in (1) by coin–tossing.

It follows that, in a randomly chosen function, the number of terms of nonlinear order i that appear is binomially distributed from 0 to $\binom{8}{i}$ with mean $\binom{8}{i}/2$. In a randomly chosen function, the number of terms of nonlinear order i that appear should be rather close to this mean.

Table 1. The number of terms of nonlinear order i, $0 \leq i \leq 8$, in the boolean functions corresponding to the eight output bits of the exponential mapping $45^{(.)}$.

order i	$\binom{8}{i}$	bit1	bit2	bit3	bit4	bit5	bit6	bit7	bit8
0	1	0	0	0	0	0	0	0	1
1	8	3	4	5	3	4	2	2	6
2	28	17	22	16	17	16	14	11	9
3	56	36	27	29	27	33	30	14	13
4	70	52	40	38	39	28	32	10	15
5	56	35	25	22	24	24	18	8	8
6	28	15	16	8	15	12	11	1	4
7	8	2	4	3	5	2	1	0	0
8	1	0	0	0	0	0	0	0	0

Table 1 shows the number of terms of each nonlinear order i that appear in the boolean function for the j-th output bit in the function $45^{(.)}$ where $j = 1$ and $j = 8$ denote the most significant and least significant bits of the output, respectively, for $i = 1, 2, \ldots, 8$. The maximum possible number of terms $\binom{8}{i}$ is also indicated. One sees immediately that, in each output bit position j, the number of terms appearing is remarkably close to the mean number $\binom{8}{i}/2$ for a randomly chosen function. Table 2 is a similar table for the function $\log_{45}(.)$ and again the agreement is remarkably close.

Table 2. The number of terms of nonlinear order i, $0 \leq i \leq 8$, in the boolean functions corresponding to the eight output bits of the logarithmic mapping $\log_{45}(.)$.

order i	$\binom{8}{i}$	bit1	bit2	bit3	bit4	bit5	bit6	bit7	bit8
0	1	1	0	0	0	0	0	0	0
1	8	5	4	4	4	0	0	0	0
2	28	12	12	13	14	22	9	11	18
3	56	26	25	27	28	24	24	38	19
4	70	39	38	34	33	43	34	26	38
5	56	25	27	18	28	29	30	19	22
6	28	13	16	15	14	17	11	9	11
7	8	1	4	2	2	3	2	1	2
8	1	0	0	0	0	0	0	0	0

It is interesting to observe that the number of terms of nonlinear order i in the least significant bit (bit 8) function for the mapping $\log_\omega(.)$ is invariant to the choice of primitive element ω in $GF(257)$. The reason is that, independently of the choice of ω, ω^k is a quadratic residue (or "square") just when k is even and hence its logarithm will have least significant bit 0 just in this case. But if ω is primitive in $GF(257)$, $\xi = \omega^i$ is also primitive if and only if i is odd. Hence any non–zero γ in $GF(257)$ is an even power of ω if and only if it is an even power of ξ and thus the least signficant bit functions in the mappings $\log_\omega(.)$ and $\log_\xi(.)$ coincide. In general, however, all the other output bit functions of the mapping $\log_\omega(.)$ and all the output bit functions of the mapping $\omega^{(.)}$ will depend on the choice of ω. However, the variation with ω is not substantial—our conclusions about Tables 1 and 2 would still apply had we chosen any other primitive element, say $\omega = 3$, of $GF(257)$ to define the exponential and logarithmic mappings and SAFER so modified would be essentially as secure as for our choice of $\omega = 45$. This choice was rather arbitrary and was motivated primarily by the apparent "randomness" in the sequence of key biases that it produces, cf. [MAS94].

4 Pseuo-Hadamard Transform

The purpose of the Pseudo-Hadamard Transform (PHT) section in Fig. 3 is to provide SAFER with *diffusion*, i.e., to ensure that small changes in round inputs cause large changes in round outputs. Because the PHT is linear over the ring of integers modulo 256 and because "differences" can be taken conveniently as byte differences modulo 256 at the output of the eight nonlinear channels in Fig. 3, diffusion is well measured by how well the PHT converts low weight inputs into high weight inputs. Here and hereafter, *weight* means the number of non-zero bytes. We now treat this question in some detail as the results are essential to the differential cryptanalysis that will be carried out in Section 6.

If the input to the PHT is the eight-byte row $\mathbf{v} = [v_1, v_2, \ldots v_8]$, then the output is the eight-byte row

$$\mathbf{V} = [V_1, V_2, \ldots V_8] = \mathbf{vM} \ ,$$

where

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

is the 8×8 matrix that we will refer to as the *PHT matrix*. The i-th row of $\mathbf{M}$ is just the PHT $\mathbf{V}$ of the input row $\mathbf{v}$ that is all-zero except in the i-th byte

where it contains a 1. From (2), the action of the PHT matrix $\mathbf{M}$ on the inputs $\mathbf{v}$ of weight 1 is evident. These results are given in Table 4 of Appendix A for outputs with weight up to 4. The only weight–1 input $\mathbf{v}$ giving an output also of weight 1 is $[128, 0, 0, 0, 0, 0, 0, 0]$ as follows from the facts that only the first row of $\mathbf{M}$ contains a single 1 and that for non–zero a $2a = 0$ if and only if $a = 128$. Similarly, it is easy to check that there are 3 different weight–1 inputs that give weight–2 outputs, none whatsoever that give weight–3 outputs, and only 5 that give weight–4 outputs. One sees from Table 4 that the PHT diffuses weight–1 inputs exceedingly well.

The situation is not so much different for weight–2 inputs. In Table 6 we list all 33 weight–2 inputs that produce a PHT of weight between 1 and 3 inclusive. In particular, we note that only three weight–2 inputs produce an output of weight 1. There are nine weight–2 inputs that produce outputs also of weight 2, the most interesting of these being $[0, 128, 0, 128, 0, 0, 0, 0]$, $[0, 0, 128, 0, 0, 0, 128, 0]$ and $[0, 0, 0, 0, 128, 128, 0, 0]$, all of which reproduce themselves. Such replicating patterns might well represent a "weakness" that one could exploit in differential cryptanalysis were it not for the fact, which will be seen in Section 6, that byte differences of 128 cannot propagate unchanged through the nonlinear section of SAFER. From Table 4 one must conclude that the PHT also diffuses weight–2 inputs admirably well.

There are roughly 2^{13} weight–2 inputs, which is a fraction about 2^{-9} of the total number of weight–2 inputs, that produce PHT outputs of weight 4. There are 9 "isolated" weight–2 inputs, listed in Table 5, that produce weight–4 outputs, but these are of little use in differential cryptanalysis because of the plethora of 128's in the output—here "isolated" refers to the fact that the only non–zero multiples of these inputs that have weight 4 and produce weight–4 outputs are the trivial multiples by 1 and -1. The remaining weight–2 inputs play a rather important role in the differential cryptanalysis of SAFER in Section 6—we call them *one-dimensional* weight–2 inputs to emphasize that they appear in sets containing all the non–zero multiples of some weight–2 input, excluding possibly the non–zero multiples by 64, 128 and -64 when these have the effect of reducing either the weight of the input or the weight of the output, or both. This makes it possible to tabulate all these inputs in a compact way as we have done in Table 7. The last entry in this table indicates, for example, that all the non–zero multiples of $[0, 0, 0, 0, 0, 0, -1, 2]$, whose PHT is $[0, 0, 1, 1, 0, 0, 1, 1]$, are weight–2 inputs, except the multiple by 128, and produce weight–4 outputs.

There are no weight–3 inputs that give a PHT of weight 1. The lists of weight–3 inputs that produce PHT outputs with weights 2 and 3 are given in Tables 9 and 10, respectively. It is evident that the PHT diffuses even weight–3 inputs very well.

We will also have use in the differential cryptanalysis of SAFER for the list of weight–4 inputs that give a PHT of weight 1. There are only five of these and they are listed in Table 8.

5 Recognition of Certain Markov Ciphers

Differential cryptanalysis, originated by Biham and Shamir [BIH90], is a general attack on iterated ciphers, i.e., on ciphers that consist of many applications in cascade of the same round function. Our discussion of differential cryptanalysis will follow the treatment in [LAI91], which introduced and exploited the notion of a Markov cipher.

Differential cryptanalysis requires that one specify a notion of difference for round inputs and round outputs. In an iterated cipher, the round input and round output must take values in the same set G. In general, one can specify the difference ΔX between two round inputs (or two round outputs) X and $X^\star$ in the manner

$$\Delta X = X \otimes (X^\star)^{-1} \tag{3}$$

where $\otimes$ is a group operation on G and where $(X^\star)^{-1}$ denotes the group inverse of $X^\star$. The cipher is then said to be a *Markov cipher* if, when the round key is chosen uniformly at random and applied to two distinct round inputs X and $X^\star$, the conditional probability $P(\Delta Y = \beta \mid \Delta X = \alpha, X = \gamma)$ for the difference of the corresponding distinct round outputs Y and $Y^\star$ is independent of γ. In other words, the conditional probability of an output difference depends only on the input difference and not on the particular value of either input. It was shown in [LAI91] that, for a Markov cipher in which the round keys are chosen independently and uniformly at random [which is the universal assumption in differential cryptanalysis], the sequence of round differences is a Markov chain for which the uniform probability distribution is a stationary distribution. It follows that if this Markov chain has a steady-state probability distribution, then this must also be the uniform distribution.

We now prove a proposition that is very useful in identifying many commonly used block ciphers as Markov ciphers.

Proposition 1. *An iterated cipher in which the round input X and round output Y take values in a set G and for which the round function has the form*

$$Y = f(S, Z_{\mathrm{b}}) \quad \textit{where} \quad S = X \otimes Z_a \ ,$$

where $\otimes$ is a group operation on G and where $Z = (Z_a, Z_{\mathrm{b}})$ is the round key, is a Markov cipher for differences defined by $\Delta X = X \otimes (X^\star)^{-1}$ and $\Delta Y = Y \otimes (Y^\star)^{-1}$. Moreover, if $\oslash$ is any other group operation on G and the output difference $\tilde{\Delta} Y$ is defined as $\tilde{\Delta} Y = Y \oslash (Y^\star)^{-I}$, where $(Y^\star)^{-I}$ is the group inverse of $Y^\star$ with respect to the group operation $\oslash$, then the conditional probability $P(\tilde{\Delta} Y = \beta \mid \Delta X = \alpha, X = \gamma)$ is also independent of γ.

Remark 1: Because a cipher must be decryptable, it follows that the function $f(S, Z_{\mathrm{b}})$ in this proposition, for every value of the partial key Z_{b}, must be an invertible function of S. No other assumption on this function is needed.
Remark 2: The latter part of the proposition, which seems unmotivated at this point, will be seen to be useful in the differential cryptanalysis of SAFER.

Proof: It suffices to prove that $P(\tilde{\Delta}Y = \beta \mid \Delta X = \alpha, X = \gamma)$ is independent of γ, since choosing $\oslash = \otimes$ implies the first claim of the proposition. To do this, we begin by noting that

$$\begin{aligned} &P(\tilde{\Delta}Y = \beta \mid \Delta X = \alpha, X = \gamma) = \\ &\sum_{\delta \in G} P(\tilde{\Delta}Y = \beta, Z_{\mathrm{a}} = \delta \mid \Delta X = \alpha, X = \gamma) = \\ &\sum_{\delta \in G} P(Z_{\mathrm{a}} = \delta \mid \Delta X = \alpha, X = \gamma) P(\tilde{\Delta}Y = \beta \mid \Delta X = \alpha, X = \gamma, Z_{\mathrm{a}} = \delta) \ . \end{aligned} \tag{4}$$

But Z_{a} is uniformly random over G and jointly independent of X and $X^\star$ so that

$$P(Z_{\mathrm{a}} = \delta \mid \Delta X = \alpha, X = \gamma) = 1/N \tag{5}$$

where N is the cardinality of G. Moreover, because $S = X \otimes Z$, it follows that

$$\Delta S = (X \otimes Z) \otimes (X^\star \otimes Z)^{-1} = X \otimes Z \otimes Z^{-1} \otimes (X^\star)^{-1} = X \otimes (X^\star)^{-1} = \Delta X$$

where we used the fact that the inverse of a group product is the product of the inverses in reverse order. Thus,

$$\begin{aligned} &P(\tilde{\Delta}Y = \beta \mid \Delta X = \alpha, X = \gamma, Z_{\mathrm{a}} = \delta) = \\ &P(\tilde{\Delta}Y = \beta \mid \Delta X = \alpha, X = \gamma, \Delta S = \alpha, S = \gamma \otimes \delta, Z_{\mathrm{a}} = \delta) = \\ &P(\tilde{\Delta}Y = \beta \mid \Delta S = \alpha, S = \gamma \otimes \delta) \end{aligned} \tag{6}$$

because, given ΔS and S, $\tilde{\Delta}Y$ has no further dependence on X and ΔX. Substituting (5) and (6) into (4) gives

$$P(\tilde{\Delta}Y = \beta \mid \Delta X = \alpha, X = \gamma) = (1/N) \sum_{\delta \in G} P(\tilde{\Delta}Y = \beta \mid \Delta S = \alpha, S = \gamma \otimes \delta) \ ,$$

which, because $\gamma \otimes \delta$ ranges over all the elements of G in this sum, is equivalent to

$$P(\tilde{\Delta}Y = \beta \mid \Delta X = \alpha, X = \gamma) = (1/N) \sum_{g \in G} P(\tilde{\Delta}Y = \beta \mid \Delta S = \alpha, S = g)$$

and hence is independent of γ, as was to be shown. □

6 Differential Cryptanalysis of SAFER

As can be seen from Fig. 3, at the beginning of a round, SAFER combines the 8–byte round input $\mathbf{X} = [X_1, X_2, \ldots X_8]$ bytewise with the 8–byte first half $\mathbf{Z}_{\mathrm{a}} = [Z_{\mathrm{a}1}, Z_{\mathrm{a}2}, \ldots Z_{\mathrm{a}8}]$ of the round key to produce the 8- byte input $\mathbf{S} = [S_1, S_2, \ldots S_8]$ to the nonlinear operations in the manner that $\mathbf{S} = \mathbf{X} \otimes \mathbf{Z}_{\mathrm{a}}$ where

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

here $\oplus$ denotes the bitwise XOR operation on bytes and $+$ denotes usual byte addition, i.e., addition modulo 256. It follows that $\otimes$ is a group operation on the set G of 8–byte words. We then obtain as an immediate consequence of Proposition 1:

Corollary 2. *SAFER is a Markov cipher when the difference ΔV between 8–byte words $\mathbf{V}$ and $\mathbf{V}^\star$ is defined in the manner $\Delta V = [V_1 \oplus V_1^\star, V_2 - V_2^\star, V_3 - V_3^\star, V_4 \oplus V_4^\star, V_5 \oplus V_5^\star, V_6 - V_6^\star, V_7 - V_7^\star, V_8 \oplus V_8^\star]$*

We now draw upon the latter part of Proposition 1 to establish a fact that will be especially useful in the differential cryptanalysis of SAFER.

Corollary 3. *When* all *output differences in SAFER are defined as byte differences modulo 256, i.e., $\tilde{\Delta} V = [V_1 - V_1^\star, V_2 - V_2^\star, V_3 - V_3^\star, V_4 - V_4^\star, V_5 - V_5^\star, V_6 - V_6^\star, V_7 - V_7^\star, V_8 - V_8^\star]$, then $P(\tilde{\Delta} Y = \beta \mid \Delta X = \alpha, X = \gamma)$ is independent of γ.*

6.1 Byte differentials and quasi-differentials

The detailed differential cryptanalysis of SAFER is facilitated by consideration of the input $\mathbf{S} = [S_1, S_2, \ldots S_8]$ to the PHT section in 3. Note that S_j is given by

$$S_j = 45^{(X_j \oplus Z_{\mathrm{a}j})} + Z_{\mathrm{b}j} \ , \qquad j \in \{1, 4, 5, 8\} \tag{7}$$

where $\mathbf{Z}_{\mathrm{a}}$ and $\mathbf{Z}_{\mathrm{b}}$ are the left and right halves of the round key, respectively. We thus refer to bytes 1, 4, 5, and 8 as the *exponential bytes*. Similarly, one notes that

$$S_j = \log_{45}(X_j + Z_{\mathrm{a}j}) \oplus Z_{\mathrm{b}j} \ , \qquad j \in \{2, 3, 6, 7\} \tag{8}$$

and we thus refer to bytes $2, 3, 6$ and 7 as the *logarithmic bytes*. We will call a pair (α, τ), considered as the value of $(\Delta X_j, \Delta S_j)$, an exponential *byte differential* for $j \in \{1, 4, 5, 8\}$ and a logarithmic byte differential for $j \in \{2, 3, 6, 7\}$. Of interest greater than that of the exponential byte differentials are the exponential byte *quasi–differentials* where the output difference is taken as the modulo 256 difference $\tilde{\Delta} S_j$ rather than as the XOR difference ΔS_j.

The principal properties of the byte differentials and quasi–differentials are summarized in Table 3. When a difference ΔV or $\tilde{\Delta} V$ is a modulo 256 difference, then interchanging the inputs X and $X^\star$ negates this difference but has no effect on differences ΔV that are XOR differences. It follows that for logarithmic byte differentials, where both input and output differences are modulo 256 differences,

$$P(\Delta S = \tau \mid \Delta X = \alpha) = P(\Delta S = -\tau \mid \Delta X = -\alpha) \ .$$

Similarly for exponential byte quasi–differentials, where only the output difference is modulo 256,

$$P(\Delta S = \tau \mid \Delta X = \alpha) = P(\Delta S = -\tau \mid \Delta X = \alpha) \ .$$

These two facts are stated in the first section of Table 3. The other entries in this table were determined by direct computation of the transition probabilities $P(\Delta S = \tau \mid \Delta X = \alpha)$ and $P(\tilde{\Delta} S = \tau \mid \Delta X = \alpha)$ with the help of (7) and (8) when the bytes $Z_{\mathrm{a}j}$ and $Z_{\mathrm{b}j}$ are chosen uniformly at random over the 256 possible byte values.

Table 3. Properties of byte differentials for SAFER.

logarithmic	exponential conventional	exponential quasi-
input difference: mod 256 output difference: mod 256	input difference: XOR output difference: XOR	input difference: XOR output difference: mod 256
$P(\tau \mid \alpha) = P(-\tau \mid -\alpha)$		$P(\tau \mid \alpha) = P(-\tau \mid \alpha)$
$P(128 \mid 128) = 0$	$P(128 \mid 128) = 0$	$P(128 \mid 128) = 0$
$P(128 \mid \alpha) = 2^{-7}$ for α odd	$P(\tau \mid 128) = 2^{-7}$ for τ odd	$P(\tau \mid 128) = 2^{-7}$ for τ odd
$P(128 \mid \alpha) = 0$ for α even		$\mathrm{avg}[P(128 \mid \alpha)] = 2^{-8.2}$ for α odd
$\max P(\tau \mid \alpha) = 2^{-6.4}$ occurs for $(\alpha, \tau) \in$ $\{(128, 48), (128, -48)\}$	$\max P(\tau \mid \alpha) = 2^{-5}$ occurs for $(\alpha, \tau) \in$ $\{(-16, 32), (103, 64),$ $(18, 128), (-108, 128),$ $(48, 128), (-78, 128),$ $(54, 128), (-115, 128),$ $(-23, 128), (102, 128),$ $(-2, 128), (103, -64)\}$	$\max P(\tau \mid \alpha) = 2^{-4.7}$ occurs for $(\alpha, \tau) \in$ $\{(79, 68), (79, -68)\ \}$
$\max P(\tau \mid 128) = 2^{-6.2}$ occurs for $\tau \in \{48, -48\}$	$\max P(128 \mid \alpha) = 2^{-5}$ occurs for $\alpha \in$ $\{18, 48, 54, 102, -115,$ $-108, -78, -23, -2\}$	$\max P(128 \mid \alpha) = 2^{-5}$ occurs for $\alpha \in$ $\{18, 48, 54, 102, -115,$ $-108, -78, -23, -2\}$

It will be convenient in the differential cryptanalysis of SAFER to have available the relations between byte differentials and byte quasi-differentials that are given in the following proposition.

Proposition 4. *For byte differences* $\Delta V = V \oplus V^\star$ *and* $\tilde{\Delta} V = V - V^\star$,
a) $\tilde{\Delta} V = 0$ *if and only if* $\Delta V = 0$;
b) $\tilde{\Delta} V = 128$ *if and only if* $\Delta V = 128$; *and*
c)$\tilde{\Delta} V$ *is odd if and only if* ΔV *is odd.*

Proof: Relation a) is trivial. Relation b) follows from the fact that $\Delta V = 128$ if and only if V and $V^\star$ differ in the most significant bit only, which is also the

necessary and sufficient condition for $\tilde{\Delta}V = 128$. Finally, ΔV is odd if and only if V and $V^\star$ differ in the least significant bit only, which is also the necessary and sufficient condition for $\tilde{\Delta}V$ to be odd. □

6.2 The PHT and byte differentials

We have already defined $\mathbf{S} = [S_1, S_2, \ldots S_8]$ as the input to the PHT section in 3. Thus, the round output $\mathbf{Y} = [Y_1, Y_2, \ldots Y_8]$ is given by $\mathbf{Y} = \mathbf{SM}$ where $\mathbf{M}$ is the PHT matrix of (2) and where all the arithmetic is modulo 256. It follows that when each component of $\Delta\mathbf{S}$ is a modulo 256 difference, i.e., when $\Delta S_j = S_j - S_j^\star$ as is the case in the logarithmic bytes and as is also the case in the exponential bytes when quasi-differences are used, then

$$\tilde{\Delta}\mathbf{Y} = \mathbf{SM} - \mathbf{S}^\star\mathbf{M} = (\tilde{\Delta}\mathbf{S})\mathbf{M} \ . \tag{9}$$

The simple relation (9) is the primary reason that it is more natural to use quasi-differentials rather than ordinary differentials in the differential cryptanalysis of SAFER.

6.3 One–round and two–round quasi-differentials

We now get to the heart of the differential cryptanalysis of SAFER, i.e., to the finding of the most probable $(r-1)$-round quasi-differentials for $r = 2, 3, \ldots$. It was shown in [LAI91] that an r-round cipher is immune from differential cryptanalysis just when all its $(r-1)$- round differentials (or quasi-differentials) are essentially equally likely. Thus, SAFER is immune from differential cryptanalysis when $(\Delta\mathbf{X}, \tilde{\Delta}\mathbf{Y}(r-1))$ takes on every possible value (α, β) with probability about $1/(2^{64}-1) \approx 2^{-64}$ when $\mathbf{X} = \mathbf{Y}(0) = \alpha$ is the plaintext and $\mathbf{Y}(i)$ is the output of the i-th round. It is convenient for a one-round quasi-differential $(\Delta\mathbf{X}(i), \tilde{\Delta}\mathbf{Y}(i))$ to consider also the PHT input $\mathbf{S}(i)$ at mid–round. To emphasize the role of $\mathbf{S}(i)$, we will write one–round quasi–differentials in *expanded view* as $(\Delta\mathbf{X}(i), \tilde{\Delta}\mathbf{S}(i), \tilde{\Delta}\mathbf{Y}(i))$. It follows from (9) that

$$\tilde{\Delta}\mathbf{Y}(i) = (\tilde{\Delta}\mathbf{S}(i))\mathbf{M}$$

where $\mathbf{M}$ is the PHT matrix of (2). The probability of the transition from $\Delta\mathbf{X}(i)$ to $\tilde{\Delta}\mathbf{Y}(i)$ is just the probability of the transition from $\Delta\mathbf{X}(i)$ to $\tilde{\Delta}\mathbf{S}(i)$ because the transition from $\tilde{\Delta}\mathbf{S}(i)$ to $\tilde{\Delta}\mathbf{Y}(i)$ is deterministic. Note that the probability of a transition from $\Delta\mathbf{X}(i)$ to $\tilde{\Delta}\mathbf{S}(i)$ is the product of the probabilities of the byte differentials (in the logarithmic bytes) and the byte quasi–differentials (in the exponential bytes) for the corresponding bytes of $\Delta\mathbf{X}(i)$ and $\tilde{\Delta}\mathbf{S}(i)$. It follows then from consideration of Table 3 that the probability of such a transition decreases as the number of bytes specified in $\tilde{\Delta}\mathbf{S}(i)$ increases, which number will generally be the same as the weight of $\Delta\mathbf{X}(i)$. Finding high probability quasi–differentials for several rounds is thus mostly a matter of finding quasi–differentials whose evolution has input differences of weight as small as possible

in every round. To a good first approximation, the probability of an i–round quasi–differential decreases as the total weight of the round inputs increases.

Table 3, which directly gives the probability of one–round byte differentials and quasi–differentials, immediately provides the justification of the following two claims in which, for brevity here and later, we have written 0^j to denote j successive zero bytes.

Claim 1 *The 1–round quasi–differential with the expanded view*

$$([79, 0^7], [68, 0^7], [32, 16, 16, -120, 16, -120, -120, 68])$$

has probability $2^{-4.7}$ *and is a most likely 1–round quasi–differential for SAFER.*

It follows from Table 3 that there are 8 such most probable quasi–differentials since any of the four exponential bytes could be chosen as the single non–zero byte and since a value of -68 in this byte of $\tilde{\Delta}\mathbf{S}(1)$ would do just as well as the value 68.

Claim 2 *The 1–round differential with the expanded view*

$$([18, 0^7], [128, 0^7], [0^7, 128])$$

has probability 2^{-5} *and is a most likely 1–round differential for SAFER.*

It follows from Table 3 that there are 48 such most probable differentials since again any of the four exponential bytes could be chosen as the single non-zero byte and since there are 12 pairs of values for these non-zero bytes of $\Delta\mathbf{X}(1)$ and $\tilde{\Delta}\mathbf{S}(1)$ that have this same maximum probability.

Claims 1 and 2 illustrate interestingly that the most likely one–round quasi–differential is slightly more probable than the most likely one–round differential, which is another argument in favor of considering the former type of 'differential' rather than the latter.

Finding the most probable two-round quasi–differential is not much more difficult.

Claim 3 *The 2–round quasi-differential* $([18, 0^7], [1, 1, 1, 1, 1, 1, 1, 1])$ *with the expanded view*
(round 1) $([18, 0^7], [128, 0^7], [0^7, 128])$
(round 2) $([0^7, 128], [0^7, 1], [1, 1, 1, 1, 1, 1, 1, 1])$
has probability 2^{-12} *and is a most likely 2–round quasi–differential for SAFER.*

This claim requires more justification. Recall from the discussion in Section 5 that differences at round inputs must be of the type $\Delta\mathbf{X}$ rather than of the type $\tilde{\Delta}\mathbf{X}$. Thus, one cannot immediately set $\tilde{\Delta}\mathbf{Y}(1)$ equal to $\Delta\mathbf{X}(2)$. However, when each component of $\tilde{\Delta}\mathbf{Y}(1)$ is either 0 or 128, it follows from Proposition 4 that this equality does hold. From Table 4, we recall that there is a unique PHT input of weight 1, namely $[128, 0^7]$, that gives an output also of weight 1, namely $[0^7, 128]$. Thus, the two-round quasi-differential in Claim 3 is the unique (up to the choice of an odd byte value for the bytes of $\Delta\mathbf{Y}(2)$, which we have arbitrarily

taken as 1) such two-round quasi-differential that has weight–1 inputs to each round—thus it has maximum probability. This probability is the product of the transition probability 2^{-5} from the 18 in the first byte (which is an exponential byte) of $\Delta\mathbf{X}(1)$ to the 128 in the first byte of $\tilde{\Delta}\mathbf{S}(1)$ and the transition probability 2^{-7} from the 128 in the eighth byte (which is also an exponential byte) of $\Delta\mathbf{X}(2)$ to the 1 in the eighth byte of $\tilde{\Delta}\mathbf{S}(2)$. There are $9\times 128 = 1152$ such most probable two-round quasi-differentials, corresponding to the 9 choices seen in Table 3 for the first byte of $\Delta\mathbf{X}(1)$ and to the 128 choices of an odd number for the eighth byte of $\tilde{\Delta}\mathbf{S}(2)$.

6.4 Three–round quasi–differentials

Finding the most probable three–round quasi–differential is a much more intricate matter. We begin by stating the solution.

Claim 4 *The 3–round quasi–differential* $([0^3, 18, 0^4], [0^3, 128, 0^4])$ *with the expanded view*
(1) $([0^3, 18, 0^4], [0^3, 128, 0^4], [0, 128, 0, 128, 0, 128, 0, 128])$
(2) $([0, 128, 0, 128, 0, 128, 0, 128], [0, b, 0, -b, 0, -b, 0, b]$*: b odd,* $[b, 0, b, 0^5])$
(3) $([c, 0, b, 0^5]$*: c odd,* $[128, 0, 128, 0^5], [0^3, 128, 0^4])$,
has probability $2^{-41.6}$ *and is a most likely 3–round quasi–differential for SAFER.*

We first show that this three–round quasi–differential has the claimed probability $2^{-41.6}$. From Table 3 we see that the transition from 18 to 128 in an exponential byte has probability 2^{-5}, which is thus the probability of the first–round transition. Because each byte of $\tilde{\Delta}\mathbf{Y}(1)$ is either 0 or 128, it follows from Proposition 4 that $\Delta\mathbf{X}(2)$ coincides with $\tilde{\Delta}\mathbf{Y}(1)$. The second round requires transitions in logarithmic bytes 2 and 6 from 128 to b and $-b$, respectively, where b can be any odd number. All byte transitions are independent because the corresponding keys for each byte are independent. A direction computation gives

$$\sum_{b\ odd} P_{\log}(b \mid 128) P_{\log}(-b \mid 128) = 2^{-7.4}$$

where $P_{\log}(b \mid a)$ is the probability of the byte quasi–differential $(\Delta X, \tilde{\Delta} S) = (a, b)$. Again from Table 3 we see that the transitions from 128 to b and $-b$ (which is also odd) in exponential bytes 4 and 8 each have probability 2^{-7}. Thus the transition in round two has probability $2^{-(7.4+7+7)} = 2^{-21.4}$. It follows further from Proposition 4 that an odd value b in exponential byte 1 of $\tilde{\Delta}\mathbf{Y}(2)$ will give an odd value c, not necessarily the same as b, in byte 1 of $\Delta\mathbf{X}(3)$. From Table 3, we see that the transition from b in (logarithmic) byte 3 of $\Delta\mathbf{X}(3)$ to 128 in byte 3 of $\Delta\mathbf{S}(3)$ has probability 2^{-7}. The probability of the transition from the odd c in exponential byte 1 of $\Delta\mathbf{X}(3)$ to 128 in byte 1 of $\tilde{\Delta}\mathbf{S}(3)$ can be well approximated by the average probability for such c, which from Table 3 is seen to be $2^{-8.2}$. Hence, the transition in round 3 has probability essentially equal to $2^{-15.2}$. The probability of the 3–round differential in the claim is thus $2^{-5} \times 2^{-21.4} \times 2^{-15.2} = 2^{-41.6}$, as was to be shown.

It is interesting to note that the above 3–round differential consists of 128 different "characteristics" [to use the language of Biham and Shamir [BIH90]], one for each odd byte value b that specifies the four non–zero bytes of $\tilde{\Delta}\mathbf{S}(2)$. An i–round characteristic is a sequence consisting of the first–round input and the outputs of rounds 1, 2, ... i. The probability of a differential is the sum of the probabilities of all the characteristics of which it is composed. It is often the case that the probability of a differential is dominated, and thus well approximated, by the probability of its most likely characteristic. However, many of its 128 characteristics contribute substantially to the probabililty of the differential in Claim 4.

We now begin the rather tedious, but essential, task of showing that the 3–round differential in Claim 4 does indeed have maximum probability. Note that the sum of the weights of the three round inputs is 7—thus our task is to show that there exists no 3–round differentials having round inputs whose weights sum to 6 or less and that any whose weights sum to 7 have probability no greater than that in Claim 4.

We begin by considering differentials for which the first–round input has weight 1. If the second–round input also has weight 1, then the second–round output must have weight 8—as follows from the proof of Claim 3—and hence the differential has very low probability. Suppose then that the second–round input has weight 2. From Table 4 we see that the two non–zero bytes must be bytes 4 and 8, or bytes 6 and 8, or bytes 7 and 8. But the third–round input cannot then have weight 1 since, by Table 6, the two non–zero bytes in the round–2 input would then have had to be bytes 1 and 2, or bytes 1 and 3, or bytes 1 and 5. Nor could the third–round input have weight 2, since Table 6 shows that the two non–zero bytes in the round–2 input would then have had to be bytes 2 and 3, or bytes 2 and 4, or bytes 2 and 5, or bytes 2 and 6, or bytes 3 and 4, or bytes 3 and 5, or bytes 3 and 7, or bytes 5 and 6, or bytes 5 and 7. Nor could the third–round input have weight 3, since Table 6 shows that the two non–zero bytes in the round–2 input would then have had to be bytes 1 and 2, or bytes 1 and 3, or bytes 1 and 4, or bytes 1 and 5, or bytes 1 and 6, or bytes 1 and 7. The third–round input can indeed have weight 4, which gives round–input weights that sum to 7, but to give larger probability than the differential in Claim 4 at least three of the non–zero bytes would have to be logarithmic bytes—Table 7 shows that all four bytes then must be logarithmic bytes (bytes 2, 3, 6 and 7) and that the two non–zero bytes in the round–2 input would have had to be bytes 2 and 5, or bytes 4 and 7, which is again a contradiction. That the second–round input cannot have weight 3 follows immediately from Table 4. Still considering a weight–1 first–round input, suppose that the second–round input has weight 4. From Table 4, these non–zero bytes must be bytes 4, 6, 7 and 8, or bytes 2, 4, 6 and 8, or bytes 3, 4, 7 and 8, or bytes 5, 6, 7 and 8. It follows then from Table 8 that the third–round input cannot have weight 1. The third–round input can indeed have weight 2, which gives round input weights that again sum to 7, but the probability of such a differential will not be larger than that of the differential in Claim 4 since only two of the four non-zero bytes in the round–2

input are logarithmic bytes. We conclude that no three–round differential with a weight–1 first–round input can have larger probability than the differential in Claim 4.

We now consider the case where the first–round input has weight 2. Suppose that the second–round input has weight 1. From Table 6 it follows that this non–zero byte must be byte 4, or byte 6, or byte 7. It then follows further from Table 4 that the input to round three must have weight at least 4—when this weight is 4, the differential is less probable than that in Claim 4 because there is no "one–dimensional" intermediate set of mid-round outputs. Suppose next that the second–round input has weight 2. It then follows from Table 6 that the two non–zero bytes in the second–round input must be bytes 2 and 4, or bytes 2 and 6, or bytes 3 and 4, or bytes 3 and 7, or bytes 4 and 6, or bytes 4 and 7, or bytes 5 and 6, or bytes 5 and 7, or bytes 6 and 7. None of these pairs can give a third–round input of weight 1 or weight 3 as follows from Table 6. Several of these pairs can be seen from Table 6 to admit third-round inputs of weight 2 but require byte transitions from 128 to 128 in the second round and hence, by Table 3, give probability 0 for the second–round transition. The second–round input can indeed have weight 4 and, in fact, the differential of Claim 4 is of this type and was chosen to give a round–3 input of weight 1 via a one–dimensional intermediate set of mid–round outputs so as to maximize its probability in this class.

We now must consider the case when the first–round input has weight 3. Table 9 shows that weight 1 is impossible for the second–round input and that weight 2 is possible only if the two non-zero bytes are bytes 2 and 8, or bytes 3 and 8, or bytes 4 and 8, or bytes 5 and 6, or bytes 6 and 8, or bytes 7 and 8. But, according to Table 6, none of these pairs can lead of a third–round input with weight less than 4. Hence, a three–round differential with first–round input of weight 3 will be much less probable than that in Claim 4.

That weight–4 first–round inputs cannot give a three–round differential with probability larger than that in Claim 4 will be evident from the treatment of 4–round differentials that follows. First–round inputs of weight 5 or more obviously need not be considered.

6.5 Four–round quasi–differentials

In light of the lengthy argument required to establish Claim 4 for three–round differentials, the reader will be pleasantly surprised to see that the four–round case follows from the former with very little additional work. In fact, the most likely four–round differential begins with the previously determined most likely three–round differential.

Claim 5 *The 4-round quasi-differential* $([0^3, 18, 0^4], [2, 1, 2, 1, 2, 1, 2, 1])$ *with the expanded view*
(1) $([0^3, 18, 0^4], [0^3, 128, 0^4], [0, 128, 0, 128, 0, 128, 0, 128])$
(2) $([0, 128, 0, 128, 0, 128, 0, 128], [0, b, 0, -b, 0, -b, 0, b]$*: b odd,* $[b, 0, b, 0^5])$
(3) $([c, 0, b, 0^5]$*: c odd,* $[128, 0, 128, 0^5], [0^3, 128, 0^4])$,
(4) $([0^3, 128, 0^4], [0^3, 1, 0^4], [2, 1, 2, 1, 2, 1, 2, 1]$

has probability $2^{-48.6}$ *and is a most likely 4–round quasi–differential for SAFER.*

The probability of the fourth–round transition is the probability of the byte quasi–differential (128, 1) [where 1 could be replaced by any odd byte value], which from Table 3 is seen to be 2^{-7}. Thus, this four–round differential has probability $2^{-41.6} \times 2^{-7} = 2^{-48.6}$ as claimed. Because the additional fourth round has a weight–1 input, essentially the same arguments as were just used for the 3–round case establish that this four–round differential likewise has maximum probability.

Note that the last three rounds of the above four–round differential constitute a three–round differential whose first–round input has weight 4. This is the most probable three-round differential of this type, but its probability $2^{-43.6}$ is smaller by a factor of 4 than the differential in Claim 4.

6.6 Five–rounds and more quasi–differentials

It is an unrewardingly tedious task to try to determine precisely the most probable differentials for SAFER for five or more rounds. The four–round differential of Claim 5 ends with a weight-8 output and hence cannot be extended with an additional round to obtain a highly probable five–round differential. Nor can an additional low–weight round be placed before these four rounds. The analysis that we have done suggests that one will need to specify at least two more byte transitions to create a good five–round differential than were necessary to specify in order to create the most likely four–round differential. One expects very conservatively that the probability of the most probable five–round differential differs by a factor of 2^{-8} [the average probability of a byte transition] or less from that of the most probably four–round differential. With virtually no doubt then, the most probable five–round differential for SAFER will have probability at most 2^{-57}. This is close enough to the average differential probability of 2^{-64} that the attack to find the key of six–round SAFER K–64 by differential cryptanalysis would require more computation than a brute–force exhaustive key search. For this reason, we abide by our original recommendation of six rounds (with a maximum of ten rounds) for SAFER K–64. For six–round SAFER K–128, however, exhaustive key search would be much more complex than the attack by differential cryptanalysis, which is why we have recommended at least ten rounds (with a maximum of twelve rounds) be used with this cipher. It could mislead users were we to allow a 128–bit key rather than a 64–bit key when the security against differential cryptanalysis would not be substantially enhanced by the longer key.

7 A Hashing 'Weakness' in SAFER

Having announced a freely available and non–proprietary cipher, we consider it our responsibility to inform present and prospective users of this cipher should any significant weaknesses be found in it. The first such 'weakness' of which

we are aware was discovered by Knudsen [KNU95] two months after the oral presentation of this paper and concerns the use of SAFER for hashing.

It is not uncommon to use secret-key ciphers within a public hashing scheme, cf. [LAI93]. The strength of the cipher for such hashing depends on the difficulty of producing 'collisions', i.e., of finding two distinct plaintext/key pairs that yield the same ciphertext. When the plaintext and ciphertext are 64 bit strings, the median number of distinct plaintext/key pairs that must be chosen uniformly at random before such a collision is found is about 2^{32}. By some very clever cryptanalysis, Knudsen devised a method to produce such collisions for *six–round* SAFER K–64 after choosing only about 2^{24} distinct plaintext/key pairs, i.e., about 256 times as fast as by random guessing. (Because SAFER K–128 reduces to SAFER K–64 when the two halves of the 128–bit key coincide, Knudsen's attack also applies to SAFER K–128.)

Knudsen exploited the fact, which can be seen from Fig. 1 for SAFER K–64, that changing one byte of the secret key K_1 changes only the byte in this same position in all $2r + 1$ round keys. This fact appears to be irrelevant for encryption because of the diffusing effect of the PHT, cf. Section 4, but it has significant implications for hashing. Two round keys differing in only one byte will sometimes encrypt a round input to the the same round output. Knudsen was able to select two secret keys differing in only one byte in such a way that both keys encrypt between 2^{22} and 2^{28} plaintexts in the same way for six rounds. This is the phenomenon that he exploited to produce collisions about 256 times faster than by random guessing when *six–round* SAFER is used within standard hashing schemes. He also found pairs of secret keys that encrypt about 2^{15} plaintexts in the same way for *eight rounds*, but this is not enough to give an advantage over random guessing in producing collisions. H also determined that there are no pairs of secret keys that encrypt many plaintexts in the same way for *ten or more rounds*.

Knudsen [KNU95] suggested a new key schedule that could be used with "SAFER" and would completely remove the hashing 'weakness' that he exploited, but that is somewhat more complicated than the original key schedules, which are described in Section 2. Although adopting Knudsen's key schedule would certainly be a more elegant cure for the hashing 'weakness' in SAFER, it seems preferable to us (in deference to the many users who have already implemented SAFER in software or in silicon) to abide by the original and simpler key schedules and merely to *specify that at least ten rounds of SAFER be used whenever SAFER is embedded in a hashing scheme* so that the hashing 'weakness' vanishes.

8 Concluding Remarks

We have attempted in the above to give a fairly complete picture of present knowledge concerning the security of SAFER. We will continue our own analysis of SAFER and will disseminate as rapidly as possible any 'weaknesses' in SAFER that we ourselves find or that are brought to our attention.

It is a pleasure here to acknowledge the contributions of the following Armenian scientists to the differential cryptanalysis of SAFER that was reported here: G. H. Khachatrian, M. K. Kuregian, and S. S. Martirossian. Their earlier studies, to which we were privy, were very helpful to us, but the responsibility for any errors in the analysis given in this paper rests of course with us.

A Tables of PHT correspondences

Table 4. Weight–1 inputs giving a PHT of weight 1, 2, 3 or 4.

input byte	input value	PHT bytes	PHT values
1	64	4 6 7 8	128 128 128 64
1	128	8	128
1	-64	4 6 7 8	128 128 128 -64
2	128	6 8	128 128
3	128	4 8	128 128
4	128	2 4 6 8	128 128 128 128
5	128	7 8	128 128
6	128	5 6 7 8	128 128 128 128
7	128	3 4 7 8	128 128 128 128

Table 5. Isolated weight–2 inputs giving a PHT of weight 4.

input bytes		input values		PHT bytes	PHT values
1	2	64	64	2 5 6 8	128 128 -64 128
1	2	-64	-64	2 5 6 8	128 128 64 128
1	3	64	64	2 3 4 8	128 128 -64 128
1	3	-64	-64	2 3 4 8	128 128 64 128
1	5	64	64	3 5 7 8	128 128 -64 128
1	5	-64	-64	3 5 7 8	128 128 64 128
2	7	128	128	3 4 6 7	128 128 128 128
3	6	128	128	4 5 6 7	128 128 128 128
4	5	128	128	2 4 6 7	128 128 128 128

Table 6. Weight–2 inputs giving a PHT of weight 1, 2 or 3.

input bytes	input values	PHT bytes	PHT values
1 2	64 128	4 7 8	128 128 -64
1 2	64 -64	2 5 6	128 128 64
1 2	128 128	6	128
1 2	-64 64	2 5 6	128 128 -64
1 2	-64 128	4 7 8	128 128 64
1 3	64 128	6 7 8	128 128 -64
1 3	64 -64	2 3 4	128 128 64
1 3	128 128	4	128
1 3	-64 64	2 3 4	128 128 -64
1 3	-64 128	6 7 8	128 128 64
1 4	64 128	2 7 8	128 128 -64
1 4	128 128	2 4 6	128 128 128
1 4	-64 128	2 7 8	128 128 64
1 5	64 128	4 6 8	128 128 -64
1 5	64 -64	3 5 7	128 128 64
1 5	128 128	7	128
1 5	-64 64	3 5 7	128 128 -64
1 5	-64 128	4 6 8	128 128 64
1 6	64 128	4 5 8	128 128 -64
1 6	128 128	5 6 7	128 128 128
1 6	-64 128	4 5 8	128 128 64
1 7	64 128	3 6 8	128 128 -64
1 7	128 128	3 4 7	128 128 128
1 7	-64 128	3 6 8	128 128 64
2 3	128 128	4 6	128 128
2 4	128 128	2 4	128 128
2 5	128 128	6 7	128 128
2 6	128 128	5 7	128 128
3 4	128 128	2 6	128 128
3 5	128 128	4 7	128 128
3 7	128 128	3 7	128 128
5 6	128 128	5 6	128 128
5 7	128 128	3 4	128 128

Table 7. One dimensional weight–2 inputs giving a PHT of weight 4.

input bytes		input values		PHT bytes				PHT values				excepting these values of a
1	2	a	-a	1	2	5	6	4a	2a	2a	a	0, 64, 128, -64
1	2	-a	2a	3	4	7	8	4a	2a	2a	a	0, 64, 128, -64
1	3	a	-a	1	2	3	4	4a	2a	2a	a	0, 64, 128, -64
1	3	-a	2a	5	6	7	8	4a	2a	2a	a	0, 64, 128, -64
1	4	-a	2a	1	2	7	8	-4a	-2a	2a	a	0, 64, 128, -64
1	5	a	-a	1	3	5	7	4a	2a	2a	a	0, 64, 128, -64
1	5	-a	2a	2	4	6	8	4a	2a	2a	a	0, 64, 128, -64
1	6	-a	2a	1	4	5	8	-4a	2a	-2a	a	0, 64, 128, -64
1	7	-a	2a	1	3	6	8	-4a	-2a	2a	a	0, 64, 128, -64
2	3	a	-a	3	4	5	6	2a	a	-2a	-a	0, 128
2	4	a	-a	1	2	3	4	2a	a	2a	a	0, 128
2	4	-a	2a	5	6	7	8	2a	a	2a	a	0, 128
2	5	a	-a	2	3	6	7	-2a	2a	-a	a	0, 128
2	6	a	-a	1	3	5	7	2a	2a	a	a	0, 128
2	6	-a	2a	2	4	6	8	2a	2a	a	a	0, 128
2	8	-a	2a	1	3	6	8	-2a	-2a	a	a	0, 128
3	4	a	-a	1	2	5	6	2a	a	2a	a	0, 128
3	4	-a	2a	3	4	7	8	2a	a	2a	a	0, 128
3	5	a	-a	2	4	5	7	-2a	-a	2a	a	0, 128
3	7	a	-a	1	3	5	7	2a	a	2a	a	0, 128
3	7	-a	2a	2	4	6	8	2a	a	2a	a	0, 128
3	8	-a	2a	1	4	5	8	-2a	a	-2a	a	0, 128
4	6	a	-a	2	4	5	7	-a	-a	a	a	0
4	7	a	-a	2	3	6	7	-a	a	-a	a	0
4	8	a	-a	1	3	5	7	a	a	a	a	0
4	8	-a	2a	2	4	6	8	a	a	a	a	0, 128
5	6	a	-a	1	2	5	6	2a	2a	a	a	0, 128
5	6	-a	2a	3	4	7	8	2a	2a	a	a	0, 128
5	7	a	-a	1	2	3	4	2a	2a	a	a	0, 128
5	7	-a	2a	5	6	7	8	2a	2a	a	a	0, 128
5	8	-a	2a	1	2	7	8	-2a	-2a	a	a	0, 128
6	7	a	-a	3	4	5	6	a	a	-a	-a	0
6	8	a	-a	1	2	3	4	a	a	a	a	0
6	8	-a	2a	5	6	7	8	a	a	a	a	0, 128
7	8	a	-a	1	2	5	6	a	a	a	a	0
7	8	-a	2a	3	4	7	8	a	a	a	a	0, 128

Table 8. Weight–4 inputs giving a PHT of weight 1.

input bytes	input values	PHT byte	PHT value
1 2 3 4	128 128 128 128	2	128
1 3 5 7	128 128 128 128	3	128
1 2 5 6	128 128 128 128	5	128
1 2 3 5	-64 128 128 128	8	64
1 2 3 5	64 128 128 128	8	-64

Table 9. Weight–3 inputs giving a PHT of weight 2. (No such inputs give a PHT of weight 1.)

input bytes	input values	PHT bytes	PHT values
1 2 3	64 128 128	7 8	128 64
1 2 3	-64 128 128	7 8	128 -64
1 2 5	64 128 128	4 8	128 64
1 2 5	-64 128 128	4 8	128 -64
1 2 7	64 128 128	3 8	128 64
1 2 7	-64 128 128	3 8	128 -64
1 3 5	64 128 128	6 8	128 64
1 3 5	-64 128 128	6 8	128 -64
1 3 6	64 128 128	5 8	128 64
1 3 6	-64 128 128	5 8	128 -64
1 4 5	64 128 128	2 8	128 64
1 4 5	-64 128 128	2 8	128 -64
2 3 4	128 128 128	2 8	128 128
2 4 6	64 128 128	6 8	64 64
2 4 6	-64 128 128	6 8	-64 -64
2 5 6	128 128 128	5 8	128 128
3 4 7	64 128 128	4 8	64 64
3 4 7	-64 128 128	4 8	-64 -64
3 5 7	128 128 128	3 8	128 128
5 6 7	64 128 128	7 8	64 64
5 6 7	-64 128 128	7 8	-64 -64

Table 10. Weight–3 inputs giving a PHT also of weight 3.

input bytes	input values	PHT bytes	PHT values
1 2 3	128 128 128	4 6 8	128 128 128
1 2 4	64 64 128	4 5 6	128 128 64
1 2 4	128 128 128	2 4 8	128 128 128
1 2 4	-64 -64 128	4 5 6	128 128 -64
1 2 5	128 128 128	6 7 8	128 128 128
1 2 6	64 64 128	2 6 7	128 64 128
1 2 6	128 128 128	5 7 8	128 128 128
1 2 6	-64 -64 128	2 6 7	128 -64 128
1 3 4	64 64 128	3 4 6	128 64 128
1 3 4	128 128 128	2 6 8	128 128 128
1 3 4	-64 -64 128	3 4 6	128 -64 128
1 3 5	128 128 128	4 7 8	128 128 128
1 3 7	64 64 128	2 4 7	128 64 128
1 3 7	128 128 128	3 7 8	128 128 128
1 3 7	-64 -64 128	2 4 7	128 -64 128
1 5 6	64 64 128	3 6 7	128 128 64
1 5 6	128 128 128	5 6 8	128 128 128
1 5 6	-64 -64 128	3 6 7	128 128 -64
1 5 7	64 64 128	4 5 7	128 128 64
1 5 7	128 128 128	3 4 8	128 128 128
1 5 7	-64 -64 128	4 5 7	128 128 -64
2 3 6	64 128 128	2 6 8	128 -64 64
2 3 6	-64 128 128	2 6 8	128 64 -64
2 3 7	128 64 128	2 4 8	128 -64 64
2 3 7	128 -64 128	2 4 8	128 64 -64
2 4 5	64 128 128	5 6 8	128 -64 64
2 4 5	-64 128 128	5 6 8	128 64 -64
2 5 7	128 64 128	5 7 8	128 -64 64
2 5 7	128 -64 128	5 7 8	128 64 -64
3 4 5	64 128 128	3 4 8	128 -64 64
3 4 5	-64 128 128	3 4 8	128 64 -64
3 5 6	128 64 128	3 7 8	128 -64 64
3 5 6	128 -64 128	3 7 8	128 64 -64

B Program for SAFER K–128

The following is a TURBO PASCAL program that implements encryption with the cipher SAFER K–128:

```
PROGRAM Full_r_Rounds_max_12_of_SAFERK128_cipher;
VAR a1,a2,a3,a4,a5,a6,a7,a8, b1,b2,b3,b4,b5,b6,b7,b8, r: byte;
  k: ARRAY[1..25,1..8] OF byte; ka, kb: ARRAY[1..8] OF byte;
  i,j,flag: integer; logtab, exptab: ARRAY[0..255] OF integer;
PROCEDURE mat1(VAR a1, a2, b1, b2: byte);
BEGIN b2:= a1 + a2; b1:= b2 + a1; END; BEGIN
{The powers of the primitive element 45 of GF(257) are computed and put in
table "exptab". Logarithms are put in table "logtab".}
  logtab[1]:= 0; exptab[0]:= 1;
  FOR i:= 1 TO 255 DO
  BEGIN
   exptab[i]:= (45 * exptab[i - 1]) mod 257; logtab[exptab[i]]:= i;
  END;
  exptab[128]:= 0; logtab[0]:= 128; exptab[0]:= 1;
  flag:= 1; writeln;
  writeln('Enter number of rounds r (max. 12) then hit CR.');
  readln(r); writeln;
  REPEAT
  BEGIN
   writeln('Enter plaintext in 8 bytes (integers from 0 to 255)');
   writeln('separated by spaces, then hit CR.');
   readln(a1, a2, a3, a4, a5, a6, a7, a8);
   writeln('Enter left half of key (Ka) in 8 bytes then hit CR.');
   readln(ka[1],ka[2],ka[3],ka[4],ka[5],ka[6],ka[7],ka[8]);
   writeln('Enter right half of key (Kb) in 8 bytes then hit CR.');
   readln(kb[1],kb[2],kb[3],kb[4],kb[5],kb[6],kb[7],kb[8]); writeln;
   writeln('Key Ka is', ka[1]:4,ka[2]:4,ka[3]:4,ka[4]:4,
      ka[5]:4,ka[6]:4,ka[7]:4,ka[8]:4);
   writeln('Key Kb is', kb[1]:4,kb[2]:4,kb[3]:4,kb[4]:4,
      kb[5]:4,kb[6]:4,kb[7]:4,kb[8]:4);
   writeln('PLAINTEXT is ',a1:8,a2:4,a3:4,a4:4,a5:4,a6:4,a7:4,a8:4);
   {The next instructions implement the key schedule that derives keys
   K1, K2, ... K2r+1 from the 128 bit input key (Ka, Kb).}
   {K1 is set equal to Kb.}
   FOR j:= 1 TO 8 DO k[1,j]:= kb[j];
   {Each byte of the key Ka is right rotated by 3.}
   FOR j:= 1 TO 8 DO ka[j]:= (ka[j] shr 3) + (ka[j] shl 5);
   FOR i:= 1 TO r DO
   BEGIN
    FOR j:= 1 TO 8 DO
    BEGIN
     {Each byte of keys Ka and Kb is further left rotated by 6.}
```

```
      ka[j]:= (ka[j] shl 6) + (ka[j] shr 2); kb[j]:= (kb[j] shl 6) + (kb[j] shr 2);
      {The key biases are added to give the keys K2i and K2i+1.}
      k[2*i,j]:= ka[j] + exptab[exptab[18*i+j]];
      k[2*i+1,j]:= kb[j] + exptab[exptab[18*i+9+j]];
    END;
  END;
  FOR i:= 1 TO r DO {The r rounds of encryption begin here.}
  BEGIN
    {Key 2i-1 is mixed bit and byte added to the round input.}
    a1:= a1 xor k[2*i-1,1]; a2:= a2 + k[2*i-1,2];
    a3:= a3 + k[2*i-1,3]; a4:= a4 xor k[2*i-1,4];
    a5:= a5 xor k[2*i-1,5]; a6:= a6 + k[2*i-1,6];
    a7:= a7 + k[2*i-1,7]; a8:= a8 xor k[2*i-1,8];
    {The result now passes through the nonlinear layer.}
    b1:=exptab[a1];b2:=logtab[a2];b3:=logtab[a3];b4:=exptab[a4];
    b5:=exptab[a5];b6:=logtab[a6];b7:=logtab[a7];b8:=exptab[a8];
    {Key 2i is now mixed byte and bit added to the result.}
    b1:= b1 + k[2*i,1]; b2:= b2 xor k[2*i,2];
    b3:= b3 xor k[2*i,3]; b4:= b4 + k[2*i,4];
    b5:= b5 + k[2*i,5]; b6:= b6 xor k[2*i,6];
    b7:= b7 xor k[2*i,7]; b8:= b8 + k[2*i,8];
    {The PHT of the result is now computed to complete the round.}
    mat1(b1, b2, a1, a2); mat1(b3, b4, a3, a4);
    mat1(b5, b6, a5, a6); mat1(b7, b8, a7, a8);
    mat1(a1, a3, b1, b2); mat1(a5, a7, b3, b4);
    mat1(a2, a4, b5, b6); mat1(a6, a8, b7, b8);
    mat1(b1, b3, a1, a2); mat1(b5, b7, a3, a4);
    mat1(b2, b4, a5, a6); mat1(b6, b8, a7, a8);
    writeln('after round',i:2,a1:8,a2:4,a3:4,a4:4,a5:4,a6:4,a7:4,a8:4);
  END;
  {Key 2r+1 is now mixed bit and byte added to form the cryptogram.}
  a1:= a1 xor k[2*r+1,1]; a2:= a2 + k[2*r+1,2];
  a3:= a3 + k[2*r+1,3]; a4:= a4 xor k[2*r+1,4];
  a5:= a5 xor k[2*r+1,5]; a6:= a6 + k[2*r+1,6];
  a7:= a7 + k[2*r+1,7]; a8:= a8 xor k[2*r+1,8];
  writeln('CRYPTOGRAM is',a1:8,a2:4,a3:4,a4:4,a5:4,a6:4,a7:4,a8:4);writeln;
  writeln('Type 1 & CR to continue, 0 & CR to stop.');readln(flag);
 END
 UNTIL flag = 0;
END.
```

C Examples of SAFER K–128 Encryption

Key Ka is	8	7	6	5	4	3	2	1
Key Kb is	8	7	6	5	4	3	2	1
after round 1	101	42	122	106	63	111	225	227
after round 2	102	122	66	171	75	196	228	30
after round 3	114	219	165	207	71	24	132	155
after round 4	117	53	164	99	161	204	201	48
after round 5	132	77	246	149	5	187	182	27
after round 6	199	89	95	137	71	106	55	152
after round 7	40	214	206	250	209	115	253	33
after round 8	166	126	11	244	39	244	4	61
after round 9	178	50	26	234	35	53	4	119
after round 10	107	97	193	179	197	19	126	173
after round 11	246	216	224	225	46	28	176	2
after round 12	47	211	218	110	13	45	17	209

Key Ka is	1	2	3	4	5	6	7	8
Key Kb is	8	7	6	5	4	3	2	1
after round 1	245	74	156	7	16	15	87	214
after round 2	154	238	95	247	240	190	143	127
after round 3	179	1	127	195	35	207	215	252
after round 4	25	120	166	188	225	251	99	51
after round 5	46	38	108	134	111	249	162	200
after round 6	130	171	126	19	101	109	29	199
after round 7	5	15	205	166	46	98	19	78
after round 8	37	162	212	102	129	250	124	2
after round 9	126	21	150	201	83	135	164	152
after round 10	204	215	66	130	100	178	191	96
after round 11	254	153	253	121	114	99	71	84
after round 12	224	39	89	225	161	235	19	140

References

[BIH90] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," in *Advances in Cryptology–CRYPT0 '90* (Eds. A. J. Menezes and S. A. Vanstone), Lecture Notes in Computer Science No. 537. Heidelberg and New York: Springer, 1991.

[BIH93] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*. New York: Springer, 1993.

[HAR95a] C. Harpes, "A Generalization of Linear Cryptanalysis Applied to SAFER," Technical Report, Signal and Info. Proc. Lab., Swiss Federal Inst. Tech., Zurich, March 9, 1995.
(`http://www.isi.ee.ethz.ch/isiworld/isi/research/`)

[HAR95b] C. Harpes, G. G. Kramer and J. L. Massey, "A Generalization of Linear Cryptanalysis and the Applicability of Matsui's Piling–Up Lemma," to be presented at EUROCRYPT '95.

[HUB90] K. Huber, "Neue Kryptographische Verfahren durch Kombination von Operationen in endlichen Körpern mit der schnellen Walshtransformation," unpublished manuscript, presented and distributed to participants at the Telesec Arbeitskreis Kryptosysteme, Darmstadt, Germany, Oct. 2, 1990.

[KNU95] L. R. Knudsen, "A Weakness in SAFER K– 64," manuscript submitted to CRYPTO '95, Feb. 16, 1995.

[LAI91] 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.

[LAI93] X. Lai and J.L. Massey, "Hash Functions Based on Block Ciphers," *Advances in Cryptology–EUROCRYPT '92* (Ed. R. A. Rueppel), Lecture Notes in Computer Science No. 658. Heidelberg and New York: Springer, 1993.

[MAS94] Massey, J. L., "SAFER K–64: A Byte-Oriented Block Ciphering Algorithm," pp. 1-17 in *Fast Software Encryption* (Ed. R. Anderson), Proceedings of the Cambridge Security Workshop, Cambridge, U. K., Dec. 9–11, 1993, Lecture Notes in Computer Science No. 809. Heidelberg and New York: Springer, 1994.

[MAT93] 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.

[MAT94] M. Matsui, "The First Experimental Cryptanalysis of the Data Encryption Standard," pp. 1–11 in *Advances in Cryptology–CRYPT0 '94* (Ed. Y. G. Desmedt), Lecture Notes in Computer Science No. 839. Heidelberg and New York: Springer, 1994.

[PER94] S. R. Perkins, "Linear Cryptanalysis of the SAFER K–64 Block Cipher," Diploma Thesis, Signal & Info. Proc. Lab., Swiss Fed. Inst. of Tech., Zurich, 15 July 1994.

[SCH92] C. P. Schnorr, "FFT–Hash II, Efficient Cryptographic Hashing," pp. 45–54 in *Advances in Cryptology–EUROCRYPT '92* (Ed. R. A. Rueppel), Lecture Notes in Computer Science No. 658. Heidelberg and New York: Springer, 1993.

[VAU95] S. Vaudenay, "On the Need of Multipermutations / Cryptanalysis of MD4 and SAFER," this volume.

This article was processed using the LaTeX macro package with LLNCS style