The Block Cipher SQUARE

Joan Daemen[1] Lars Knudsen[2] Vincent Rijmen[⋆2]

[1] Banksys
Haachtesteenweg 1442
B-1130 Brussel, Belgium
Daemen.J@banksys.be
[2] Katholieke Universiteit Leuven , ESAT-COSIC
K. Mercierlaan 94, B-3001 Heverlee, Belgium
lars.knudsen@esat.kuleuven.ac.be
vincent.rijmen@esat.kuleuven.ac.be

Abstract. In this paper we present a new 128-bit block cipher called SQUARE. The original design of SQUARE concentrates on the resistance against differential and linear cryptanalysis. However, after the initial design a dedicated attack was mounted that forced us to augment the number of rounds. The goal of this paper is the publication of the resulting cipher for public scrutiny. A C implementation of SQUARE is available that runs at 2.63 MByte/s on a 100 MHz Pentium. Our M68HC05 Smart Card implementation fits in 547 bytes and takes less than 2 msec. (4 MHz Clock). The high degree of parallellism allows hardware implementations in the Gbit/s range today.

1 Introduction

In this paper, we propose the block cipher SQUARE. It has a block length and key length of 128 bits. However, its modular design approach allows extensions to higher block lengths in a straightforward way. The cipher has a new self-reciprocal structure, similar to that of THREEWAY and SHARK [2, 15].

The structure of the cipher, i.e., the types of building blocks and their interaction, has been carefully chosen to allow very efficient implementations on a wide range of processors. The specific choice of the building blocks themselves has been led by the resistance of the cipher against differential and linear cryptanalysis. After treating the structure of the cipher and its consequences for implementations, we explain the strategies followed to thwart linear and differential cryptanalysis. This is followed by a description of an efficient attack that exploits the particular properties of the cipher structure.

We do not encourage anyone to use SQUARE today in any sensitive application. Clearly, confidence in the security of any cryptographic design must be based on the resistance against effective cryptanalysis after intense public scrutiny.

⋆ F.W.O research assistant, sponsored by the Fund for Scientific Research – Flanders (Belgium).

A reference implementation of SQUARE is available from the following URL: http://www.esat.kuleuven.ac.be/~rijmen/square.

2 Structure of SQUARE

SQUARE is an iterated block cipher with a block length and a key length of 128 bits each. The round transformation of SQUARE is composed of four distinct transformations. It is however important to note that these four building blocks can be efficiently combined in a single set of table-lookups and exor operations. This will be treated later in the section on implementation aspects.

The basic building blocks of the cipher are five different invertible transformations that operate on a 4×4 array of bytes. The element of a state a in row i and column j is specified as $a_{i,j}$. Both indexes start from 0.

2.1 A Linear Transformation θ

θ is a linear operation that operates separately on each of the four rows of a state. We have

$$\theta : b = \theta(a) \Leftrightarrow b_{i,j} = c_j a_{i,0} \oplus c_{j-1} a_{i,1} \oplus c_{j-2} a_{i,2} \oplus c_{j-3} a_{i,3},$$

where the multiplication is in GF(2^8) and the indices of c must be taken modulo 4. Note that the field GF(2^n) has characteristic 2 [9]. This means that the addition in the field corresponds to the bitwise exor.

The rows of a state can be denoted by polynomials, i.e., the polynomial corresponding to row i of a state a is given by

$$a_i(x) = a_{i,0} \oplus a_{i,1} x \oplus a_{i,2} x^2 a_{i,3} x^3.$$

Using this notation, and defining $c(x) = \bigoplus_j c_j x^j$ we can describe θ as a modular polynomial multiplication:

$$b = \theta(a) \Leftrightarrow b_i(x) = c(x) a_i(x) \bmod 1 \oplus x^4 \qquad \text{for} \qquad 0 \leq i < 4.$$

The inverse of θ corresponds to a polynomial $d(x)$ given by

$$d(x)c(x) = 1 \pmod{1 \oplus x^4}.$$

2.2 A Nonlinear Transformation γ

γ is a nonlinear byte substitution, identical for all bytes. We have

$$\gamma : b = \gamma(a) \Leftrightarrow b_{i,j} = \mathrm{S}_\gamma(a_{i,j}),$$

with S_γ an invertible 8-bit substitution table or S-box. The inverse of γ consists of the application of the inverse substitution S_γ^{-1} to all bytes of a state.

2.3 A Byte Permutation π

The effect of π is the interchanging of rows and columns of a state. We have

$$\pi : b = \pi(a) \Leftrightarrow b_{i,j} = a_{j,i}.$$

π is an involution, hence $\pi^{-1} = \pi$.

2.4 Bitwise Round Key Addition σ

$\sigma[k^t]$ consists of the bitwise addition of a round key k^t. We have

$$\sigma[k^t] : b = \sigma[k^t](a) \Leftrightarrow b = a \oplus k^t.$$

The inverse of $\sigma[k^t]$ is $\sigma[k^t]$ itself.

2.5 The Round Key Evolution ψ

The round keys k^t are derived from the cipher key K in the following way. k^0 equals the cipher key K. The other round keys are derived iteratively by means of the invertible affine transformation ψ.

$$\psi : k^t = \psi(k^{t-1})$$

2.6 The Cipher Square

The building blocks are composed into the round transformation denoted by $\rho[k^t]$:

$$\rho[k^t] = \sigma[k^t] \circ \pi \circ \gamma \circ \theta \tag{1}$$

Square is defined as eight rounds preceeded by a key addition $\sigma[k^0]$ and by θ^{-1}:

$$\textsc{Square}[k] = \rho[k^8] \circ \rho[k^7] \circ \rho[k^6] \circ \rho[k^5] \circ \rho[k^4] \circ \rho[k^3] \circ \rho[k^2] \circ \rho[k^1] \circ \sigma[k^0] \circ \theta^{-1} \tag{2}$$

2.7 The Inverse Cipher

As will be shown in Section 9, the structure of Square lends itself to efficient implementations. For a number of modes of operation it is important that this is also the case for the inverse cipher. Therefore, Square has been designed in such a way that the structure of its inverse is equal to that of the cipher itself, with the exception of the key schedule. Note that this identity in *structure* differs from the identity of *components and structure* in IDEA [10].

From (2) it can be seen that

$$\begin{aligned}\textsc{Square}^{-1}[k] = \theta \circ \sigma^{-1}[k^0] \circ \rho^{-1}[k^1] \circ \rho^{-1}[k^2] \circ \rho^{-1}[k^3] \circ \rho^{-1}[k^4] \circ \\ \rho^{-1}[k^5] \circ \rho^{-1}[k^6] \circ \rho^{-1}[k^7] \circ \rho^{-1}[k^8]\end{aligned}$$

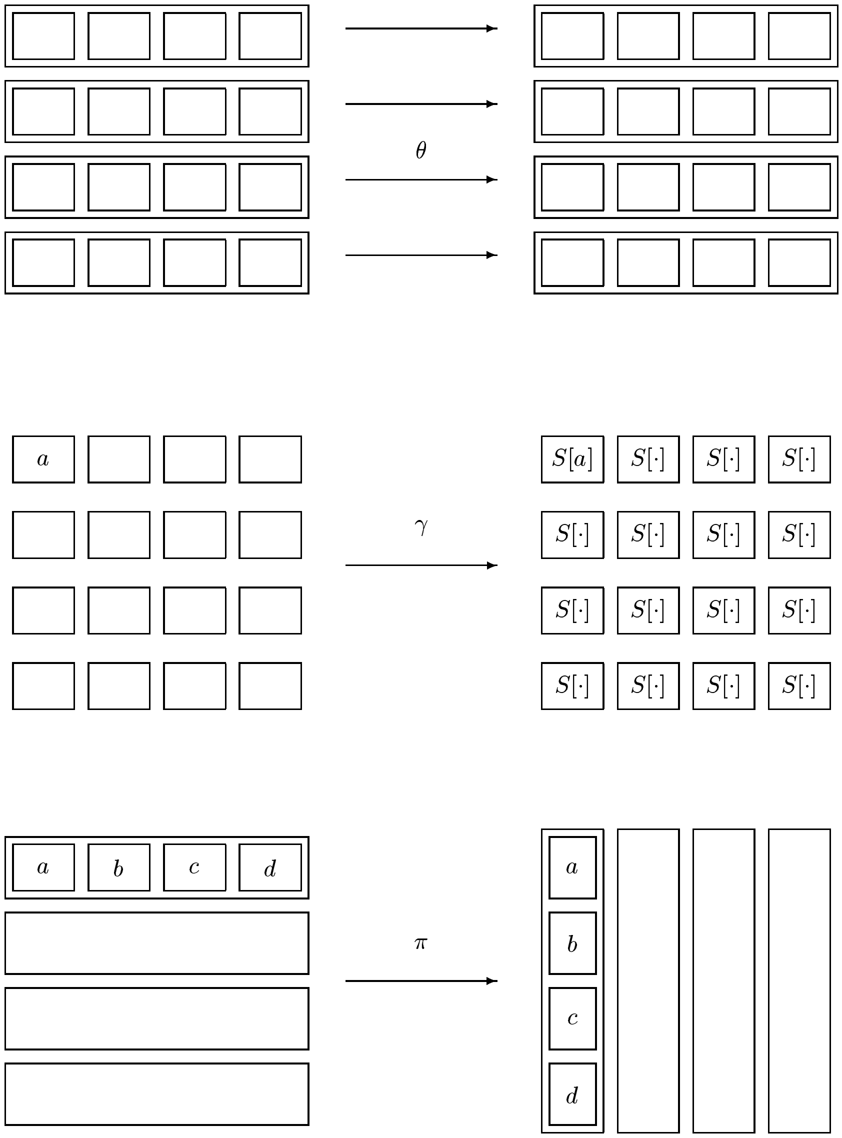

Fig. 1. Geometrical representation of the basic operations of SQUARE. θ consists of 4 parallel linear diffusion mappings. γ consists of 16 separate substitutions. π is a transposition.

with

$$\rho^{-1}[k^t] = \theta^{-1} \circ \gamma^{-1} \circ \pi^{-1} \circ \sigma^{-1}[k^t] = \theta^{-1} \circ \gamma^{-1} \circ \pi \circ \sigma[k^t] \tag{3}$$

It may seem that the structure of the inverse cipher differs substantially from that of the cipher itself. By exploiting some algebraic properties of the building blocks, we can show this not to be the case. Since π only transposes the bytes $a_{i,j}$ and γ only operates on the individual bytes, independent of their position (i, j), we have

$$\gamma^{-1} \circ \pi = \pi \circ \gamma^{-1}.$$

Moreover, since $\theta^{-1}(a) \oplus k^t = \theta^{-1}(a + \theta(k^t))$, we have

$$\sigma[k^t] \circ \theta^{-1} = \theta^{-1} \circ \sigma[\theta(k^t)],$$

We now define the round transformation of the inverse cipher as

$$\rho'[k^t] = \sigma[k^t] \circ \pi \circ \gamma^{-1} \circ \theta^{-1}, \tag{4}$$

which has the same structure as ρ itself, except that γ and θ are replaced by γ^{-1} and θ^{-1} respectively. Using the algebraic properties above, we can derive

$$\begin{aligned}
\theta \circ \sigma[k^0] \circ \rho^{-1}[k^1] &= \theta \circ \sigma[k^0] \circ \theta^{-1} \circ \gamma^{-1} \circ \pi \circ \sigma[k^1] \\
&= \theta \circ \theta^{-1} \circ \sigma[\theta(k^0)] \circ \pi \circ \gamma^{-1} \circ \sigma[k^1] \\
&= \sigma[\theta(k^0)] \circ \pi \circ \gamma^{-1} \circ \sigma[k^1] \\
&= \sigma[\theta(k^0)] \circ \pi \circ \gamma^{-1} \circ \sigma[k^1] \circ \theta^{-1} \circ \theta \\
&= \sigma[\theta(k^0)] \circ \pi \circ \gamma^{-1} \circ \theta^{-1} \circ \sigma[\theta(k^1)] \circ \theta \\
&= \rho'[\theta(k^0)] \circ \sigma[\theta(k^1)] \circ \theta
\end{aligned}$$

This equation can be generalized in a straightforward way to include more than one round. Now, with $\kappa^t = \theta(k^{8-t})$, we have

$$\begin{aligned}
&\text{SQUARE}^{-1} = \\
&\rho'[\kappa^8] \circ \rho'[\kappa^7] \circ \rho'[\kappa^6] \circ \rho'[\kappa^5] \circ \rho'[\kappa^4] \circ \rho'[\kappa^3] \circ \rho'[\kappa^2] \circ \rho'[\kappa^1] \circ \sigma[\kappa^0] \circ \theta
\end{aligned}$$

Hence the inverse cipher is equal to the cipher itself with γ replaced by γ^{-1}, with θ by θ^{-1} and different round key values.

2.8 First round

The θ^{-1} before $\sigma[k^0]$ in SQUARE can be incorporated in the first round. We have

$$\begin{aligned}
\rho[k^1] \circ \sigma[k^0] \circ \theta^{-1} &= \sigma[k^1] \circ \pi \circ \gamma \circ \theta \circ \sigma[k^0] \circ \theta^{-1} \\
&= \sigma[k^1] \circ \pi \circ \gamma \circ \sigma[\theta(k^0)]
\end{aligned}$$

Hence the initial θ^{-1} can be discarded by omitting θ in the first round and applying $\theta(k^0)$ instead of k^0. The same simplification can be applied to the inverse cipher.

3 Linear and Differential Cryptanalysis

The resistance against linear cryptanalysis [12] and differential cryptanalysis [1] has been the rationale behind the criteria by which the S_γ substitution and the θ multiplication polynomial $c(x)$ have been chosen.

A difference propagation along the rounds of an iterated block cipher is generally called a differential characteristic. A characteristic is specified by a series of difference patterns. The probability associated with a characteristic is the probability that all intermediate difference patterns have the value specified in the above series. We call a differential characteristic a differential *trail*. The probability associated with a differential trail can be approximated by the product of the difference propagations between every pair of subsequent rounds (which can be easily calculated). The probability that a given difference pattern a' at the input of a number of cipher rounds gives rise to a difference pattern b' at the output is equal to the sum of the probabilities of all differential trails starting with a' and ending with b'. In general the propagation from the input difference pattern a' to the output difference pattern b' is called a *differential*.

As can be seen in [12] the correlation between a linear combination of input bits and a linear combination of output bits of an iterated block cipher can be treated in an analogous but slightly different way. A *linear trail* is specified by a series of selection patterns. For a given cipher key, the *correlation coefficient* (positive or negative) corresponding to a linear trail consists of the product of the correlation coefficients between the linear combinations of bits of every pair of subsequent rounds. In [2] it was shown that the correlation between a linear combination of input bits, denoted by selection pattern u, and a linear combination of output bits, denoted by v is equal to the sum of the correlation coefficients of all linear trails starting with u and ending in v. It must be remarked that the correlation coefficients may be positive or negative and that the sign depends on the value of round key bits.

S_γ and $c(x)$ are chosen to minimize the maximum probability of differential trails and the maximum correlation of linear trails over four rounds. This is obtained in the framework of a very specific approach.

3.1 Wide Trail Design Strategy

In [2] the 'wide trail design strategy' was introduced as a means to guarantee low maximum probability of multiple-round differential trails and low maximum correlation of multiple-round linear trails. In this strategy the round transformation is composed of a number of uniform transformations, that are split in the nonlinear blockwise substitution (corresponding to our γ) and the composition of the linear transformations (corresponding to our $\theta \circ \pi$). The round key addition does not play a role in the strategy. It was shown in [2] that the probability of a differential trail is the product of the input-output difference propagation probabilities of the S-boxes with nonzero input difference ('active S-boxes'). The correlation of a linear trail is the product of the input-output correlations of the S-boxes with nonzero output selection patterns ('active S-boxes'). The two mechanisms for eliminating high-probability differential trails and high-correlation linear trails are the following:

- Choose an S-box where the maximum difference propagation probability and the maximum input-output correlation are as small as possible.
- Choose the linear part in such a way that there are no trails with few active S-boxes.

The first mechanism gives us two clear criteria for the selection of the S_γ. The second mechanism gives a hint on how to select the multiplication polynomial $c(x)$. In the following section we will focus on the linear part $\theta \circ \pi$.

4 The Multiplication Polynomial $c(x)$

The transformation θ treats the different rows of a state a completely separately and in the same way. We will now study the difference propagation and correlation properties of θ, concentrating on a single row. Assume an input difference specified by $a'(x) = a(x) \oplus a^*(x)$. The output difference will be given by

$$b'(x) = c(x)a(x) \oplus c(x)a^*(x) \bmod 1 \oplus x^4 = c(x)a'(x) \bmod 1 \oplus x^4.$$

On the other hand, a linear combination of output bits, specified by the selection polynomial $u(x)$ is equal to (i.e., correlated to, with correlation coefficient 1) a linear combination of input bits, specified by the following selection polynomial [2]:

$$v(x) = c(x^{-1})u(x) \bmod 1 + x^4.$$

It is intuitively clear that both linear and differential trails would benefit from a multiplication polynomial that could limit the number of nonzero terms in input and output difference (and selection) polynomials. This is exactly what we want to avoid by choosing a polynomial with a high diffusion power, expressed by the so-called *branch number*.

Let $w_h(a)$ denote the Hamming weight of a vector, i.e., the number of nonzero components in that vector. Applied to a state a, a difference pattern a' or a selection pattern u, this corresponds to the number of non-zero bytes. In [2] the branch number $\mathcal{B}$ of an invertible linear mapping was introduced as

$$\mathcal{B}(\theta) = \min_{a \neq 0}(w_h(a) + w_h(\theta(a))) .$$

This implies that the sum of the Hamming weights of a pair of input and output difference patterns (or selection patterns) to θ is at least $\mathcal{B}$. It can easily be shown that $\mathcal{B}$ is a lower bound for the number of active S-boxes in two consecutive rounds of a linear or differential trail. Since θ operates on each row separately, we can have $\mathcal{B} = 5$ at most.

In [15] it was shown how a linear mapping over $\mathrm{GF}(2^m)^n$ with optimal $\mathcal{B}$ ($\mathcal{B} = n + 1$) can be constructed from a maximum distance separable code. The

MDS-code used is a Reed-Solomon code over GF(2^m): if $G_e = [I_{n\times n} B_{n\times n}]$ is the echelon form of the generation matrix of and $(2n, n, n+1)$-RS-code, then $\theta : X \mapsto Y = B \cdot X$ defines a linear mapping with optimal branch number.

The polynomial multiplication with $c(x)$ corresponds to a special subset of the MDS-codes, having the additional property that B is a circulant matrix. A circulant matrix is a matrix where every row consists of the same elements, shifted over one position, or $b_{i,j} = b_{0,j-i \bmod n}$. This property is exploited in section 9.2 to produce a memory-efficient implementation of the cipher. In [11] we find the following theorem:

Theorem 1. *An (n, k, d)-code $\mathcal{C}$ with generator matrix $G = [IB]$ is MDS iff every square submatrix of B is nonsingular.*

In a matrix with elements from GF(2^m) every determinant has a probability of 2^{-m} to evaluate to zero. For increasing size of the matrix the number of determinants increases exponentially, making it infeasible to search randomly for an MDS-code. However, in a circulant matrix the number of distinct determinants is only a fraction of the number for arbitrary matrices (cf. Table 1). By imposing the extra constraint that the matrix should be a circulant, we increase the probability to find an MDS-code by random search.

n	generic	circulant	n	generic	circulant
1	1	1	5	252	41
2	5	3	6	924	111
3	20	7	7	3431	309
4	70	17	8	12869	935

Table 1. The number of square submatrices in a generic matrix of order n, and the number of non-equivalent determinants in a circulant matrix of the same order. The numbers of the last column were obtained by an exhaustive computer search.

$c(x)$ corresponds to a 4×4 matrix, hence if we choose it randomly, the probability that it has $\mathcal{B} = 5$ can be approximated by $(1 - \frac{1}{256})^{17} \approx 0.93$. This gives us a high degree of freedom in the choice of $c(x)$. We choose

$$c(x) = \mathtt{2_x} \oplus \mathtt{1_x} \cdot x \oplus \mathtt{1_x} \cdot x^2 \oplus \mathtt{3_x} \cdot x^3 .$$

This determines $d(x)$ uniquely.

$$d(x) = \mathtt{E_x} \oplus \mathtt{9_x} \cdot x \oplus \mathtt{D_x} \cdot x^2 \oplus \mathtt{B_x} \cdot x^3$$

4.1 Motivation for the Choice of π

Since the branch number of $c(x)$ is 5, the number of active S-boxes in a two-round trail is at least 5. The effect of π, interchanging rows and columns, has the effect that any trail over four consecutive rounds will have at least 25 active S-boxes. A simple and clear proof of this is available and will be published in a more theoretical paper that is being written [3] .

5 The Nonlinear Substitution γ

As explained above, the relevant criteria imposed upon the γ S-box are the highest (in absolute value) occurring correlation between any pair of linear combination of input bits and linear combinations of output bits (denoted by λ) and the highest occurring probability corresponding to any pair of input difference and output difference pattern. This corresponds to the highest value in the so-called exor table of the γ S-box, defined as

$$E_{ij} = \#\{x|S(x) \oplus S(x \oplus i) = j\} .$$

We define $\delta = \max_{i,j}\{E_{ij}\} \cdot 2^{-8}$.

We present three alternative choices for the S-box: explicitly constructed nonlinear algebraic transformations, slightly modified versions of the latter and randomly selected invertible mappings.

5.1 Explicit Construction

In [13] a method is given to construct m-bit S-boxes with $\gamma = 2^{1-m/2}$ and $\delta = 2^{2-m}$, the theoretically minimum possible values. From the proposals in [13] we select the mapping $x \mapsto x^{-1}$ over $\mathrm{GF}(2^8)$, with $\delta = 2^{-6}$ and $\lambda = 2^{-3}$.

The problem with this choice is that the mapping has a very simple description in $\mathrm{GF}(2^8)$. The other components of the round transformation also have a simple description in $\mathrm{GF}(2^8)$. This may enable cryptanalytic attacks based on the algebraic manipulation of equations to derive key information [4].

Note that any m-bit mapping can be represented as a polynomial or a rational form in $\mathrm{GF}(2^m)$. It is however unlikely that this representation can be exploited in cryptanalysis if the polynomial or rational form is of no special, relatively simple, form.

The feasibility of algebraic manipulation can be severely diminished. The elements of $\mathrm{GF}(2^8)$ can be represented with respect to different bases. By choosing a different basis for the definition of θ and γ we can prevent that the round transformation has a simple description in any basis of $\mathrm{GF}(2^8)$.

Still, even specified in another basis, the chosen nonlinear mapping stays an involution and has two fixed points: 0 and 1. By applying an affine transformation on the individual bits of the output these properties can be removed and a simple algebraic expression of the round transformation in any basis of $\mathrm{GF}(2^8)$ can be prevented.

5.2 Modifications

Another method to prevent a simple algebraic description is by choosing a mapping according to the method explained in the previous subsection and subsequently modifying it slightly to destroy the exploitable algebraic structure. It will be seen that the disadvantage of this approach is that δ and/or λ will increase.

We conducted some experiments starting from the mapping multiplicative inverse in GF(2^8) as proposed above ($\delta = 2^{-6}$ and $\lambda = 8 \times 2^{-6}$) and we applied a small number of modifications.

When we consider the mapping as a look-up table and investigate all variants that have a pair of entries swapped, an increase is observed of δ to $6 \cdot 2^{-8}$ and/or λ to 9×2^{-6}. We also tested 300 000 variants in which four or eight entries were swapped. Swapping four entries increases λ to 9×2^{-6}, swapping eight entries increases λ to 10×2^{-6} and δ to $6 \cdot 2^{-8}$.

5.3 Random Search

Algebraically constructed permutations always exhibit some structure that may be exploited in attacks in unanticipated ways, designers often resort to random substitutions: a substitution is selected from a set of substitutions that are generated by the use of a random source and evaluated with respect to (presumably) relevant nonlinearity criteria. In [14] the average differential properties of permutations are investigated and a bound for the expected value of δ is given. For an m-bit permutation

$$\lim_{m \to \infty} \frac{E[\delta 2^m]}{2m} \leq 1\,.$$

We verified this experimentally for 1.5 million samples with $m = 8$ and measured at the same time δ and λ. The results are given in table 2. The S-boxes with the highest resistance against both linear and differential cryptanalysis, have $\delta = 10 \cdot 2^{-8}$ and $\lambda = 15 \cdot 2^{-6}$.

λ	δ						
	$8 \cdot 2^{-8}$	$10 \cdot 2^{-8}$	$12 \cdot 2^{-8}$	$14 \cdot 2^{-8}$	$16 \cdot 2^{-8}$	$18 \cdot 2^{-8}$	$20 \cdot 2^{-8}$
15×2^{-6}	0	0.07	0.07	0.006	0.0001	0	0
16×2^{-6}	0.0003	4.77	5.58	0.58	0.04	0.002	0
17×2^{-6}	0.002	15.63	20.55	2.24	0.15	0.007	0.0004
18×2^{-6}	0.0002	12.21	17.17	1.96	0.13	0.007	0.0005
19×2^{-6}	0.0004	4.91	7.31	0.87	0.05	0.003	0
20×2^{-6}	0	1.52	2.34	0.28	0.02	0.001	0
21×2^{-6}	0	0.41	0.64	0.08	0.004	0.001	0

Table 2. Maximum input-output correlation and difference propagation probability of randomly generated nonlinear permutations. The entries denote the percentage of the generated mappings that have the indicated λ and δ.

5.4 Our Choice

Because of its optimal values for λ and δ, we have decided to take for S_γ an S-box that is constructed by taking the mapping $x \mapsto x^{-1}$ and applying an affine transformation (over GF(2)) to the output bits. This affine transformation has the property that it has a complicated description in $\mathrm{GF}(2^8)$ to thwart interpolation attacks [4].

Our choices force all four-round differential trails to have an associated probability not higher than 2^{-150}, far below the critical noise value of 2^{-127}. Equivalently, four-round linear trails have an associated correlation not over 2^{-75}, far below the critical noise value of 2^{-64}. Hence, for resistance against conventional LC and DC six rounds may seem sufficient. However, the specific blocked structure of the cipher allows for more efficient dedicated differential attacks. This will be explained in the following section.

6 A Dedicated Attack

In this section we describe a dedicated attack that exploits the cipher structure of SQUARE. The attack is a chosen plaintext attack and is independent of the specific choices of S_γ, $c(x)$ and the key schedule. It is faster than an exhaustive key search for SQUARE versions of up to 6 rounds. After describing the basic attack on 4 rounds, we will show how it can be extended to 5 and 6 rounds.

6.1 Preliminaries

Let a Λ-set be a set of 256 states that are all different in some of the (16) state bytes (the *active*) and all equal in the other state bytes (the *passive*). Let λ be the set of indices of the active bytes. We have

$$\forall x, y \in \Lambda : \begin{cases} x_{i,j} \neq y_{i,j} \text{ for } (i,j) \in \lambda \\ x_{i,j} = y_{i,j} \text{ for } (i,j) \notin \lambda \end{cases}$$

In this section we will make use of the geometrical interpretation as presented in Figure 1. Applying the transformations γ and $\sigma[k^t]$ on (the elements of) a Λ-set results in a (generally different) Λ-set with the same λ. Applying π results in a Λ-set in which the active bytes are transposed by π. Applying θ to a Λ-set does not necessarily result in a Λ-set. However, since every output byte of γ is a linear combination (with invertible coefficients) of the four input bytes in the same row, an input row with a single active byte gives rise to an output row with only active bytes.

6.2 Four Rounds

Consider a Λ-set in which only one byte is active. We will now trace the evolution of the positions of the active bytes through 3 rounds. The 1st round contains no θ, hence there is still only one byte active at the beginning of the 2nd round. θ of the

2nd round converts this to a complete row of active bytes, that is subsequently transformed by π to a complete column. θ of the 3rd round converts this to a Λ-set with only active bytes. This is still the case at the input to the 4th round.

Since the bytes of the outputs of the 3rd round (denoted by a) range over all possible values and are therefore balanced over the λ-set, we have

$$\bigoplus_{b=\theta(a), a \in \Lambda} b_{i,j} = \bigoplus_{a \in \Lambda} \bigoplus_{k} c_{j-k} a_{i,k} = \bigoplus_{l} c_l \bigoplus_{a \in \Lambda} a_{i,l+j} = \bigoplus_{l} c_l 0 = 0.$$

Hence, the bytes of the output of θ of the fourth round are balanced. This balancedness is in general destroyed by the subsequent application of γ.

An output byte of the 4th round (denoted by a here) can be expressed as a function of the intermediate state b above

$$a_{i,j} = S_\gamma[b_{j,i}] \oplus k^4_{i,j}.$$

By assuming a value for $k^4_{i,j}$, the value of $b_{j,i}$ for all elements of the Λ-set can be calculated from the ciphertexts. If the values of this byte are not balanced over Λ, the assumed value for the key byte was wrong. This is expected to eliminate all but approximately 1 key value. This can be repeated for the other bytes of k^4.

We implemented the attack and found that two Λ-sets of 256 chosen plaintexts each are sufficient to uniquely determine the cipher key with an overwhelming probability of success.

6.3 Extension by a Round at the End

If an additional round is added, we have to calculate the above value of $b_{j,i}$ from the output of the 5th round instead of the 4th round. This can be done by additionally assuming a value for a set of 4 bytes of the 5th round key. As in the case of the 4-round attack, wrong key assumptions are eliminated by verifying that $b_{j,i}$ is not balanced.

In this 5-round attack 2^{40} key values must be checked, and this must be repeated 4 times. Since by checking a single Λ-set leaves only $1/256$ of the wrong key assumptions as possible candidates, the cipher key can be found with overwhelming probability with only 5 Λ-sets.

6.4 Extension by a Round at the Beginning

The basic idea is to choose a set of plaintexts that results in a Λ-set at the output of the 2nd round with a single active S-box. This requires the assumption of values of four bytes of the round key k^0.

If the intermediate state after θ of the 2nd round has only a single active byte, this is also the case for the output of the 2nd round. This imposes the following conditions on a row of four input bytes of θ of the second round: one particular linear combination of these bytes must range over all 256 possible values (active) while 3 other particular linear combinations must be constant for all 256 states. This imposes identical conditions on the bytes in the same row in the input to $\sigma[k^1]$, and consequently on a column of bytes in the input to π of the 1st round. If the corresponding column of bytes of k^0 is known, these conditions can be converted to conditions on four plaintext bytes.

Now we consider a set of 2^{32} plaintexts, such that the array of bytes in one column ranges over all possible values and all other bytes are constant.

Now, make an assumption for the value of the 4 bytes of the relevant column of k^0. Select from the set of 2^{32} available plaintexts, a set of 256 plaintexts that obey the conditions indicated above. Now the 4-round attack can be performed. For the given key assumption, the attack can be repeated for a several plaintext sets. If the byte values of k^5 suggested by these attacks are not consistent, the initial assumption must have been wrong. A correct assumption for the bytes of k^0 will result in the swift and consistent recuperation of the last round key.

We implemented this attack where we assumed knowledge of 16 bits of the first-round key. The attack found the other 16 bits of the first-round key and 128 bits of the last-round key using only 2 structures of 256 plaintexts for every key value guessed in the first round.

6.5 Complexity of the Attacks

Combining both extensions results in a 6 round attack. Although infeasible with current technology, this attack is faster than exhaustive key search, and therefore relevant. We have not found extensions to 7 rounds faster than exhaustive key search.

We summarize the attacks in Table 3.

Attack	#Plaintexts	Time	Memory
4-round	2^9	2^9	small
5-round type 1	2^{11}	2^{40}	small
5-round type 2	2^{32}	2^{40}	2^{32}
6-round	2^{32}	2^{72}	2^{32}

Table 3. Complexities of the attack on SQUARE.

7 Number of Rounds

Due to these attacks we have to increase the number of rounds to at least seven. As a safety margin, we fixed the number of rounds to eight.

Conservative users are free to increase the number of rounds. This can be done in a straightforward way and requires no adaptation of the key schedule whatsoever.

8 The Key Evolution ψ

The key schedule specifies the derivation of the round keys in terms of the cipher key. Its function is to provide resistance against the following types of attack:

- Attacks in which part of the cipher key is known to the cryptanalyst, e.g., if the cipher is used with a key shorter than 128 bits.
- Attacks where the key entry to the cipher is known or can be chosen, e.g., if the cipher is used as the compression function of a hash algorithm [7].
- Related-key attacks.

Resistance against the first type of attack can be improved by a key schedule in which the round key undergoes a transformation with high diffusion. For a good scheme, the knowledge of a certain number of bits of one round key fixes very few bits in other round keys. The other two types of attack exploit regularities in the structure of the key schedule by locally compensating round key differences [5, 7].

The key schedule also plays an important role in the elimination of symmetry:

- **Symmetry in the round transformation**: the round transformation treats all bytes of a state in very much the same way. This symmetry can be removed by having round constants in the key schedule.
- **Symmetry between the rounds**: the round transformation is the same for all rounds. This equality can be removed by having round-dependent round constants in the key schedule.

The key schedule is defined in terms of the rows of the key. We can define a left byte-rotation operation $\mathsf{rotl}(a_i)$ on a row as

$$\mathsf{rotl}[a_{i,0}a_{i,1}a_{i,2}a_{i,3}] = [a_{i,1}a_{i,2}a_{i,3}a_{i,0}]$$

and a right byte rotation $\mathsf{rotr}(a_i)$ as its inverse.

The key schedule iteration transformation $k^{t+1} = \psi(k^t)$ and its inverse are defined by

$$\begin{aligned} k_0^{t+1} &= k_0^t \oplus \mathsf{rotl}(k_3^t) \oplus C_t \\ k_1^{t+1} &= k_1^t \oplus k_0^{t+1} \\ k_2^{t+1} &= k_2^t \oplus k_1^{t+1} \\ k_3^{t+1} &= k_3^t \oplus k_2^{t+1} \end{aligned} \qquad \begin{aligned} \kappa_3^{t+1} &= \kappa_3^t \oplus \kappa_2^t \\ \kappa_2^{t+1} &= \kappa_2^t \oplus \kappa_1^t \\ \kappa_1^{t+1} &= \kappa_1^t \oplus \kappa_0^t \\ \kappa_0^{t+1} &= \kappa_0^t \oplus \mathsf{rotr}(\kappa_3^t) \oplus C'_t \end{aligned}$$

The simplicity of the inverse key schedule is thanks to the fact that θ and ψ commute. The round constants C_t are also defined iteratively. We have $C_0 = \mathtt{1_x}$ and $C_t = \mathtt{2_x} \cdot C_{t-1}$.

This choice provides high diffusion and removes the regularities in an efficient way.

9 Implementation Aspects

9.1 8-bit Processor

On an 8-bit processor SQUARE can be programmed by simply implementing the different component transformations. This is straightforward for π, σ and ψ. The transformation γ requires a table of 256 bytes. θ requires multiplication in the field $\mathrm{GF}(2^8)$. However, the multiplication polynomial has been chosen to make this very efficient. We have written a program implementing SQUARE in Assembler for the Motorola's M68HC05 microprocessor, typical for Smart Cards. The machine code occupies in total 547 bytes of ROM, needs 36 bytes of RAM and one execution of SQUARE, including the key schedule, takes about 7500 cycles. This corresponds to less than 2 msec with a 4 MHz Clock.

The inverse cipher however is significantly slower than the forward cipher. This is caused by the difference in complexity between θ and θ^{-1}.

9.2 32-bit Processor

In the implementation of the cipher, the succession of steps

$$\theta \circ \sigma[k^t] \circ \pi \circ \gamma = \sigma[k'^t] \circ \theta \circ \pi \circ \gamma$$

with $k'^t = \theta(k^t)$ can be combined in a single set of table lookups. The intermediate state can be represented by four 32-bit words, each containing a row $[a_i]$. Its transpose is denoted by $[a_i]^T$. For $b = \theta(\pi(\gamma(a))) + k'^t$ we have

$$\begin{aligned}
[b_i]^T &= \begin{bmatrix} c_0 & c_3 & c_2 & c_1 \\ c_1 & c_0 & c_3 & c_2 \\ c_2 & c_1 & c_0 & c_3 \\ c_3 & c_2 & c_1 & c_0 \end{bmatrix} \cdot \begin{bmatrix} S_\gamma[a_{0,i}] \\ S_\gamma[a_{1,i}] \\ S_\gamma[a_{2,i}] \\ S_\gamma[a_{3,i}] \end{bmatrix} \oplus [k_i'^t]^T \\
&= \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \cdot S_\gamma[a_{0,i}] \oplus \begin{bmatrix} c_3 \\ c_0 \\ c_1 \\ c_2 \end{bmatrix} \cdot S_\gamma[a_{1,i}] \oplus \begin{bmatrix} c_2 \\ c_3 \\ c_0 \\ c_1 \end{bmatrix} \cdot S_\gamma[a_{2,i}] \oplus \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_0 \end{bmatrix} \cdot S_\gamma[a_{3,i}] \oplus [k_i'^t]^T
\end{aligned}$$

We define the tables M and T as

$$\begin{aligned}
M[a] &= a \cdot \begin{bmatrix} c_0 & c_1 & c_2 & c_3 \end{bmatrix} \\
T[a] &= M[S[a]] \,.
\end{aligned}$$

T and M have 256 entries of four bytes each. The table M implements the polynomial multiplication. T combines the nonlinear substitution with this multiplication. Now we have

$$[b_i] = \bigoplus_j \mathsf{rotr}^j (T[a_{ji}]) \oplus [k_i'^t] \,.$$

We conclude that $\sigma[\theta(k_i^t)] \circ \theta \circ \pi \circ \gamma$ can be done with 16 table lookups, 12 rotations and 16 exors of 32-bit words. This implementation needs the table T, with 256 entries of four bytes, i.e. one kilobyte in total.

Last Round It can be seen that in this implementation, θ of the last round is already executed in the previous set of table-lookups. In the last round the function to be applied is $\sigma[k^8] \circ \pi \circ \gamma$. This can be realised by replacing the table $T[x] = M[S[x]]$ by $S[x]$. Since $c_2 = \mathtt{1_x}$, the unity in $\mathrm{GF}(2^8)$, the entries of the small table S can be extracted from T, removing the extra storage requirement for S.

Performance The reference implementation is written in C and runs at 2.63 MByte/s on a 100 MHz Pentium with the Windows95 operating system. The inverse cipher can be implemented in exactly the same way as the cipher itself and has the same performance. The difference is in the tables and the precalculation of the round keys.

10 Acknowledgements

We thank Paulo Barreto, who wrote the optimized reference implementation of SQUARE. Paulo Barreto can be reached at `pbarreto@uninet.com.br`.

References

1. E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptology*, Vol. 4, No. 1, 1991, pp. 3–72.
2. J. Daemen, "Cipher and hash function design strategies based on linear and differential cryptanalysis," *Doctoral Dissertation*, March 1995, K.U.Leuven.
3. J. Daemen and V. Rijmen, "Self-reciprocal cipher structures," *COSIC internal report 96-3*, 1996.
4. T. Jakobsen and L.R. Knudsen, "The interpolation attack on block ciphers," *these proceedings.*
5. J. Kelsey, B. Schneier and D. Wagner, "Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES," *Advances in Cryptology, Proceedings Crypto'96, LNCS 1109*, N. Koblitz, Ed., Springer-Verlag, 1996, pp. 237–252.
6. L.R. Knudsen, "Truncated and higher order differentials," *Fast Software Encryption, LNCS 1008*, B. Preneel, Ed., Springer-Verlag, 1995, pp. 196–211.
7. L.R. Knudsen, "A key-schedule weakness in SAFER-K64," *Advances in Cryptology, Proceedings Crypto'95, LNCS 963*, D. Coppersmith, Ed., Springer-Verlag, 1995, pp. 274–286.
8. L.R. Knudsen and T.A. Berson, "Truncated differentials of SAFER," *Fast Software Encryption, LNCS 1039*, D. Gollmann, Ed., Springer-Verlag, 1996, pp. 15–26.
9. N. Koblitz, *"A Course in Number Theory and Cryptography,"* Springer-Verlag, New York, 1987.
10. X. Lai, J.L. Massey and S. Murphy, "Markov ciphers and differential cryptanalysis," *Advances in Cryptology, Proceedings Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 17–38.
11. F.J. MacWilliams, N.J.A. Sloane, *"The Theory of Error-Correcting Codes,"* North-Holland, Amsterdam, 1977.

12. M. Matsui, "Linear cryptanalysis method for DES cipher," *Advances in Cryptology, Proceedings Eurocrypt'93, LNCS 765*, T. Helleseth, Ed., Springer-Verlag, 1994, pp. 386–397.
13. K. Nyberg, "Differentially uniform mappings for cryptography," *Advances in Cryptology, Proceedings Eurocrypt'93, LNCS 765*, T. Helleseth, Ed., Springer-Verlag, 1994, pp. 55–64.
14. L. O'Connor, "On the distribution of characteristics in bijective mappings," *Journal of Cryptology*, Vol. 8, No. 2, 1995, pp. 67–86..
15. V. Rijmen, J. Daemen et al., "The cipher SHARK," *Fast Software Encryption, LNCS 1039*, D. Gollmann, Ed., Springer-Verlag, 1996, pp. 99–112.