A Chosen Plaintext Attack of the 16-round Khufu Cryptosystem

Henri Gilbert and Pascal Chauvaud

France Télécom-CNET
PAA-TSA-SRC
38-40 Rue du Général Leclerc, 92131 Issy-les-Moulineaux
France

Abstract. In 1990, Merkle proposed two fast software encryption functions, Khafre and Khufu, as possible replacements for DES [1]. In 1991, Biham and Shamir applied their differential cryptanalysis technique to Khafre [2], and obtained an efficient attack of the 16-round version and some bounds on the 24-round version. However, these attacks take advantage of the fact that the S-boxes used for Khafre are public; they cannot be applied to Khufu, which uses secret S-boxes, and no attack of Khufu has been proposed so far. In this paper, we present a chosen plaintext attack of the 16-round version of Khufu, which is based on differential properties of this algorithm. The derivation of first information concerning the secret key requires about 2^{31} chosen plaintexts and 2^{31} operations. Our estimate of the resources required for breaking the entire scheme is about 2^{43} chosen plaintexts and about 2^{43} operations.

1 Description of Khufu

Khufu is an iterated blockcipher with a 64-bit blocksize. The keys used in the 16-round version (the single one considered in this paper) are the following :

- four 32-bit words K1, K2, K3, K4, used before the first round and after the last round (initial and final xors);
- four secret permutations p_0, p_1, p_2 and p_3 of the [0..255] set (the columns of the first S-box introduced in [1]), which provide the functions used in rounds 1 to 8;
- four secret permutations q_0, q_1, q_2 and q_3 of the [0..255] set (the columns of the second S-box introduced in [1]), which provide the functions used in rounds 9 to 16.

The 16-round version of Khufu is depicted in Figure 1, which represents the encryption of one 64-bit block consisting of two 4-byte halves $L = (l_0,l_1,l_2,l_3)$ and $R = (r_0,r_1,r_2,r_3)$. We are using the following notations :

- π_i (i = 0 to 3) denotes the projection :
  $\pi_i : [0..255]^4 \rightarrow [0..255]$
  $(x_0, x_1, x_2, x_3) \mapsto x_i$

Y.G. Desmedt (Ed.): Advances in Cryptology - CRYPTO '94, LNCS 839, pp. 359-368, 1994.

– $[p_{i_0}, p_{i_1}, p_{i_2}, p_{i_3}]$ (where (i_0, i_1, i_2, i_3) is a circular permutation of (0,1,2,3)) denotes an S-box the columns of which are provided by the $p_{i_0}, p_{i_1}, p_{i_2}$ and p_{i_3} permutations :

$$[p_{i_0}, p_{i_1}, p_{i_2}, p_{i_3}] : [0..255] \rightarrow [0..255]^4$$
$$x \mapsto (p_{i_0}[x], p_{i_1}[x], p_{i_2}[x], p_{i_3}[x])$$

The representation of Figure 1 is slightly different from the one provided in [1]: in order to avoid swapping the right and left halves R and L and rotating the R half at each round, we are using a different round function at each round. It is however easy to check that both representations are strictly equivalent.
For a more detailed presentation of the Khufu algorithm, see [1].

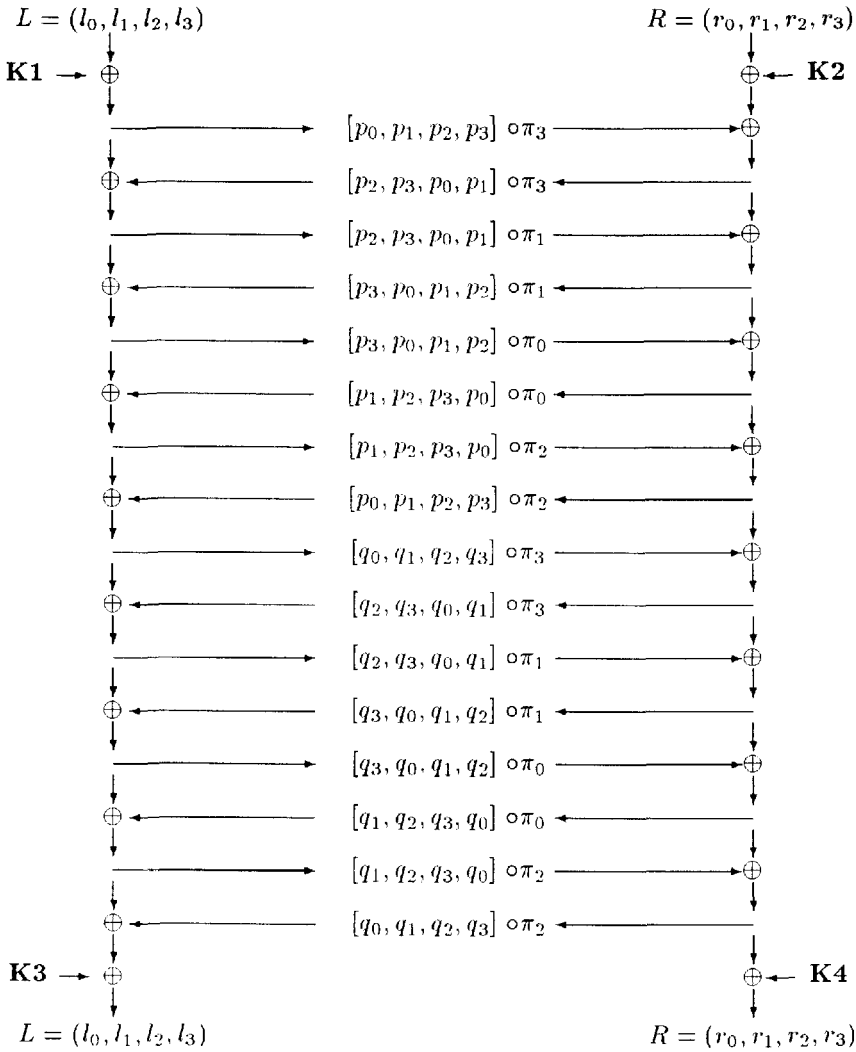

Fig. 1. The 16-round version of Khufu

2 Differential properties used in our attack

At the various steps of our attack, sets of n (P,P') pairs of 64-bit plaintext blocks are considered. In practice, these sets are defined by a combination of conditions on :

- the plaintext difference P $\oplus$ P';
- some bytes of P.

Let S be such a set of n (P,P') pairs. After r rounds (where $r \in [0,..,16]$), a (P,P') pair provides a $(C[r], C'[r])$ pair of intermediate blocks, of xor value : $\Delta[r] = C[r] \oplus C'[r]$. We denote by $\Delta S[r]$ the set of $\Delta[r]$ values derived from the n (P,P') pairs in S after r rounds.

All the differential properties we are using in our attack can be expressed in terms of the cardinal $\mid \Delta S[r] \mid$ of the $\Delta S[r]$ set of the distinct difference values after r rounds for an appropriately selected set S of plaintext pairs.

The following Proposition will be useful for the initial step of our attack:

Proposition 1. *Let* λ_1, λ_3 *and* ρ_3 *be arbitrary constants in* [0..255].
For every $\alpha \in]0..255]$, *there exist four constants* δ_0, δ_1, δ_2 *and* $\beta \in [0..255]$ *such that, if* $S_{\lambda_1,\lambda_3,\rho_3,\alpha}$ *denotes the following set of plaintext pairs :*
$S_{\lambda_1,\lambda_3,\rho_3,\alpha} = \{((L,R);(L',R')) \mid L \oplus L' = (0,\alpha,0,0);\ R \oplus R' = (\delta_0,\delta_1,\delta_2,0);$
$l_1 = \lambda_1;\ l_3 = \lambda_3;\ r_3 = \rho_3\}$
then $\Delta S_{\lambda_1,\lambda_3,\rho_3,\alpha}[9] = ((0,\alpha,0,0),(0,0,0,\beta))$,
i. e. the difference for the various pairs of the $S_{\lambda_1,\lambda_3,\rho_3,\alpha}$ *set is constant after 9 rounds.*

Proof : After two rounds, the (l_1, l'_1) pair is fixed and equal to the two constant bytes :
$c_1 = \lambda_1 \oplus\ K1_1\ \oplus\ p_3\ [\rho_3 \oplus K2_3\ \oplus\ p_3[\lambda_3 \oplus K1_3]]$; $c'_1 = c_1 \oplus \alpha$;

furthermore, these l_1 and l'_1 values act as inputs to the $[p_2, p_3, p_0, p_1]$ S-box at round 3. It suffices to select $\delta_0, \delta_1, \delta_2$ as to "compensate" the difference between the two $[p_2, p_3, p_0, p_1]$ outputs to obtain a constant xor value for the six subsequent rounds. The encryption of a $S_{\lambda_1,\lambda_3,\rho_3,\alpha}$ pair is depicted in Figure 2 (where the difference value at each round is provided).
The $\delta_0, \delta_1, \delta_2$ and β constants are given by the relation :
(1) : $(\delta_0, \delta_1, \delta_2, \beta) = [p_2, p_3, p_0, p_1][c_1]\ \oplus\ [p_2, p_3, p_0, p_1][c'_1]$ where c_1 and c'_1 are defined above.

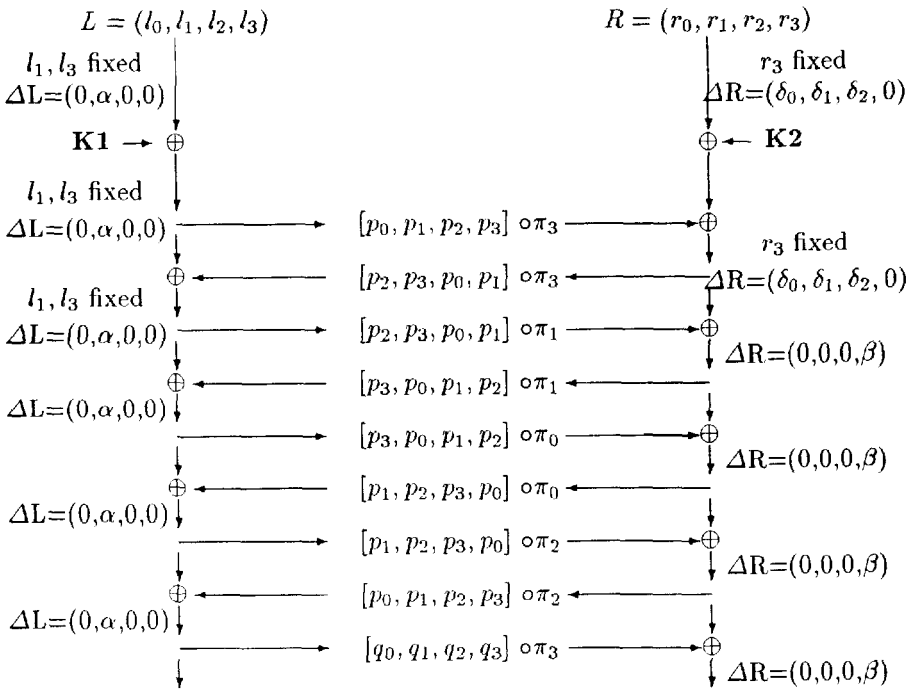

Fig. 2. Encryption of an $S_{\lambda_1,\lambda_3,\rho_3,\alpha}$ pair

The next proposition, which is very similar to Proposition 1, will be useful for the subsequent steps of our attack.

Proposition 2. *Let λ_3 be arbitrary constant $\in$ [0..255]. For every $\alpha \in$]0..255], there exist four constants $\delta_0, \delta_1, \delta_2$ and $\delta_3 \in$ [0..255] such that, if $S_{\lambda_3,\alpha}$ denotes the following set of plaintext pairs :*
$S_{\lambda_3,\alpha} = \{((L,R);(L',R')) \mid L \oplus L' = (0,0,0,\alpha); R \oplus R' = (\delta_0,\delta_1,\delta_2,\delta_3); l_3 = \lambda_3\}$ *then* $\Delta S_{\lambda_3,\alpha}[8] = \{((0,0,0,\alpha),(0,0,0,0))\}$, *i. e. the difference for the various pairs of the $S_{\lambda_3,\alpha}$ set is constant after 8 rounds.*

Proof : The proof is similar to the proof of Proposition 1. The l_3 and l'_3 inputs to the $[p_0,p_1,p_2,p_3]$ S-box in the first round are the constant bytes:
$c_3 = \lambda_3 \oplus K1_3$ and $c'_3 = c_3 \oplus \alpha$.
It suffices to take for $\delta_0, \delta_1, \delta_2$ and δ_3 the constants given by the relation :
(2) : $(\delta_0, \delta_1, \delta_2, \delta_3) = [p_0, p_1, p_2, p_3]\ [c_3] \oplus [p_2, p_3, p_0, p_1]\ [c'_3]$ to obtain the announced result.

The next proposition essentially states that for a given set of plaintext pairs, there are some bounds limiting the increase of the number of distinct difference values after r rounds. When used in conjonction with Propositions 1 and 2, Proposition 3 provides non-trivial differential properties for the whole 16-round algorithm.

Proposition 3. *If S is a set of n (P,P') plaintext pairs and if $\Delta S[r]$ is defined as above, then :*
$\forall\, r \in [0..15],\ |\Delta S[r+1]| \leq 128 \times |\Delta S[r]|$

Proof : Let $\Delta[r+1]$ be a $\Delta S[r+1]$ value. There exists a (P,P') pair in S leading to the $\Delta[r+1]$ xor value after $r+1$ rounds. Let us denote by $\Delta[r]$ the xor value for this pair after r rounds, and by $\delta[r]$ the byte of $\Delta[r]$ which position is picked up as an input to the S-box used in round $(r+1)$; by a the input to the S-box of round $(r+1)$ in the encryption of P.
Depending on the parity of r, we have either :

$(*) : \Delta[r+1] = \Delta[r] \oplus ((Sbox[a] \oplus Sbox[a \oplus \delta[r]]), (0,0,0,0))$
or
$(**) : \Delta[r+1] = \Delta[r] \oplus ((0,0,0,0), (Sbox[a] \oplus Sbox[a \oplus \delta[r]]))$.

Moreover, $\Delta[r] \in \Delta S[r]$ by definition.

The results now follow from the obvious property that, for a fixed value of $\Delta[r]$ (which determines a fixed value for $\delta[r]$), we have :
$|\ \{\ Sbox[a] \oplus Sbox[a \oplus \delta[r]]\ |\ a \in [0..255]\ \}\ | \leq 128$.

3 An attack of the 16-round Khufu

In this Section, we describe how to use the differential properties presented above for deriving the secret key (i.e. four 32-bit words and 8 permutations of the [0..255] set) from about 2^{43} chosen plaintexts.

The proposed attack is divided in four main steps. Only the first Step (the derivation of first information about the secret key) is developed in some detail. The purpose of the description of the subsequent steps is only to give some evidence that the information obtained at Step 1 can be used for breaking the entire scheme.

Step 1 is based on Propositions 1 and 3. We are using the notation of Proposition 1. We fix four arbitrary constants λ_1, λ_3 and ρ_3 and $\alpha \neq 0$ in [0..255].
The purpose of Step 1 is to find the three corresponding bytes $\delta_0, \delta_1, \delta_2$.
We select a fixed arbitrary byte λ_2. For 64 different values of l_0 (for instance the 0 to 63 values) we perform the following computations :

• We encrypt the X_{l_0} set of 2^{24} plaintext values $((l_0, \lambda_1, \lambda_2, \lambda_3), (r_0, r_1, r_2, \rho_3))$, where $(r_0, r_1, r_2) \in [0..255]^3$, thus obtaining 2^{24} encrypted values $((ll_0, ll_1, ll_2, ll_3), (rr_0, rr_1, rr_2, rr_3))$;

• We encrypt the X'_{l_0} set of 2^{24} plaintext values $((l_0, \lambda_1 \oplus \alpha, \lambda_2, \lambda_3), (r'_0, r'_1, r'_2, \rho_3))$ where $(r'_0, r'_1, r'_2) \in [0..255]^3$; thus obtaining 2^{24} encrypted values $((ll'_0, ll'_1, ll'_2, ll'_3), (rr'_0, rr'_1, rr'_2, rr'_3))$;

• We now want to find all the (P,P') pairs in $(X_{l_0} \times X'_{l_0})$ such that $ll_2 = ll'_2$; $rr_0 = rr'_0$; $rr_2 = rr'_2$ (i.e. we want to filter those pairs of $(X_{l_0} \times X'_{l_0})$ for which the inputs and outputs of the S-boxes in the three last rounds of the encryption are equal), and group these pairs according to the $(r_0 \oplus r'_0, r_1 \oplus r'_1, r_2 \oplus r'_2)$ difference value. This can be done efficiently, in about 2^{25} operations in average, as follows :
- we group the X_{l_0} plaintexts according to the (ll_2, rr_0, rr_2) value, thus obtaining a list of X_{l_0} plaintexts of average size 1 for each (ll_2, rr_0, rr_2) triplet;
- we group the X'_{l_0} plaintexts according to the (ll'_2, rr'_0, rr'_2) value, thus obtaining a list of X'_{l_0} plaintexts of average size 1 for each (ll'_2, rr'_0, rr'_2) triplet;
- for each (ll_2, rr_0, rr_2), we consider all the (P,P') pairs in the crossproduct of the corresponding lists of X_{l_0} and X'_{l_0} (average number of pairs : 1);
- we group the (P,P') pairs obtained as above according to the $(r_0 \oplus r'_0, r_1 \oplus r'_1, r_2 \oplus r'_2)$ difference, thus obtaining a list of average size 1 for each $(r_0 \oplus r'_0, r_1 \oplus r'_1, r_2 \oplus r'_2)$ value.

For each $(r_0 \oplus r'_0, r_1 \oplus r'_1, r_2 \oplus r'_2)$ triplet, we thus obtain, by merging the lists obtained for each l_0 value, a list S of 64 (P,P') pairs in average. Let us consider the ΔS set of output xors for the (P,P') pairs contained in such a list :

Claim :

(i) : If $(r_0 \oplus r'_0, r_1 \oplus r'_1, r_2 \oplus r'_2) = (\delta_0, \delta_1, \delta_2)$ *then* $|\Delta S| \leq 16$.

(ii) : If $(r_0 \oplus r'_0, r_1 \oplus r'_1, r_2 \oplus r'_2) \neq (\delta_0, \delta_1, \delta_2)$ *then* $|\Delta S| \approx 64$.

Proof (heuristic arguments) : In the first case, S is a subset of the $S_{\lambda_1, \lambda_3, \rho_3, \alpha}$ considered in Proposition 1. Moreover, $\Delta S = \Delta S[13]$, because the three last rounds have no effect on the output xor of an S pair and $|\Delta S[13]| \leq |\Delta S_{\lambda_1, \lambda_3, \rho_3, \alpha}[13]|/2^{24}$ because $\Delta S[13]$ contains only $\Delta S_{\lambda_1, \lambda_3, \rho_3, \alpha}[13]$ elements $((\delta_{l_0}, \delta_{l_1}, \delta_{l_2}, \delta_{l_3}), (\delta_{r_0}, \delta_{r_1}, \delta_{r_2}, \delta_{r_3}))$ such that $\delta_{l_2} = 0$ and $\delta_{r_0} = 0$ and $\delta_{r_2} = 0$.
The first part of the claim now results from :
$|\Delta S_{\lambda_1, \lambda_3, \rho_3, \alpha}[9]| = 1$ (by Proposition 1)
and
$|\Delta S_{\lambda_1, \lambda_3, \rho_3, \alpha}[13]| \leq 128^4 |\Delta S_{\lambda_1, \lambda_3, \rho_3, \alpha}[9]|$ (by Proposition 3).
The second part of the claim follows from the assumption that in other cases,

the final xor values for pairs in the S set simply behave as random elements of the set of $((\delta_{l_0}, \delta_{l_1}, \delta_{l_2}, \delta_{l_3}), (\delta_{r_0}, \delta_{r_1}, \delta_{r_2}, \delta_{r_3}))$ differences such that $\delta_{l_2} = 0$ and $\delta_{r_0} = 0$ and $\delta_{r_2} = 0$.
The above claim was confirmed by computer experiments, where (i) was checked. Because of the difference between the behaviours *(i)* and *(ii)*, the $(\delta_0, \delta_1, \delta_2)$ triplet can be detected, as the one leading to a ΔS set of output ***xors*** of size less than 16.

Step 1 requires the encryption of about 2^{31} chosen plaintext blocks, and about 2^{31} operations.

Step 2 : we repeat Step 1 for all possible $\alpha \neq 0$ values (without modifying the λ_1, λ_3 and ρ_3 constants). We are using the notation of Proposition 1 : c_1 denotes the common input to the S-box of round 3 for all the plaintexts of the various X_{l_0} sets.
For every α, we obtain the 24-bit word :
$(\delta_0, \delta_1, \delta_2)(\alpha) = [p_2, p_3, p_0][c_1] \oplus [p_2, p_3, p_0][c_1 \oplus \alpha]$.

In summary, after Step 2, the p_2, p_3 and p_0 permutations are entirely determined, up to the four unknown bytes c_1, $p_2[c_1]$, $p_3[c_1]$ and $p_0[c_1]$. Steps 1 and 2 require to encrypt about 2^{39} plaintext blocks, and the computational cost of steps 1 and 2 is about 2^{39} operations.

Step 3 is based on Propositions 2 and 3. We fix an arbitrary constant $\lambda_3 \in [0..255]$. We are using the notations of Proposition 2. The purpose of Step 3 is to find for every $\alpha \in [1..255]$ the four corresponding bytes δ_0, δ_1, δ_2 and δ_3, such that :

$(\delta_0, \delta_1, \delta_2, \delta_3)(\alpha) = [p_0, p_1, p_2, p_3][c_3] \oplus [p_0, p_1, p_2, p_3][c_3 \oplus \alpha]$.

We are using the fact that since $[p_2,p_3,p_0]$ is known up to the unknown bytes $c_1, p_2[c_1], p_3[c_1]$ and $p_0[c_1], \delta_0, \delta_2$, and δ_3 are known up to the single unknown byte $c_1 \oplus c_3$: there are only 256 possible values for the $(\delta_0, \delta_2, \delta_3)$ triplet.

Step 3 is divided in two substeps, numbered 3.1 and 3.2.

Substep 3.1 : We want to find $(\delta_0, \delta_1, \delta_2, \delta_3)$ for a first fixed $\alpha \neq 0$ value. We are doing an exhaustive search on the $c_1 \oplus c_3$ byte. For each $c_1 \oplus c_3$ assumption, the candidate values δ_0, δ_2 and δ_3 are determined, and we efficiently test the 256 candidate δ_1 values by a method close to the one of Step 1. For that purpose, we select fixed arbitrary bytes ρ_0, ρ_2 and ρ_3; for 2^{24} different values of the (l_0, l_1, l_2) triplet, we perform the following computations :

• We encrypt the $X_{l_0 l_1 l_2}$ set of 2^8 plaintext values $((l_0, l_1, l_2, \lambda_3), (\rho_0, r_1, \rho_2, \rho_3))$, where $(r_1 \in [0..255])$, thus obtaining 2^8 encrypted values

$((ll_0, ll_1, ll_2, ll_3), (rr_0, rr_1, rr_2, rr_3))$.(Note: this can be done once for all for all $c_1 \oplus c_3$ assumptions).

• We encrypt the $X'_{l_0 l_1 l_2}$ set of 2^8 plaintext values $((l_0, l_1, l_2, \lambda_3 \oplus \alpha), (\rho_0 \oplus \delta_0, r'_1, \rho_2 \oplus \delta_2, \rho_3 \oplus \delta_3))$ where $r'_1 \in [0..255]$; thus obtaining 2^8 encrypted values $((ll'_0, ll'_1, ll'_2, ll'_3), (rr'_0, rr'_1, rr'_2, rr'_3))$;

• We now want to find all the (P,P') pairs in $(X_{l_0 l_1 l_2} \times X'_{l_0 l_1 l_2})$ such that $ll_2 = ll'_2$; $rr_2 = rr'_2$ (i.e. we want to "filter" those pairs of $(X_{l_0 l_1 l_2} \times X'_{l_0 l_1 l_2})$ for which the inputs and outputs of the S-boxes in the two last rounds of the encryption are equal), and group these pairs according to the $r_1 \oplus r'_1$ difference value. This can be done efficiently, in about 2^8 operations in average, and provides in average one (P,P') pair after filtering.

For each $\delta_1 = r_1 \oplus r'_1$ candidate difference, we thus obtain, by merging the contributions obtained for each $l_0 l_1 l_2$ value, a list S of 2^{16} (P,P') pairs in average. Let us consider the ΔS set of output xors for the (P,P') pairs contained in such a list :

Claim :

(i) : If the $(\delta_0, \delta_1, \delta_2, \delta_3)$ candidate is correct, then the ΔS differences are picked from a set of size less than $128^6 / 256^2$.

(ii) : If the $(\delta_0, \delta_1, \delta_2, \delta_3)$ candidate is wrong, then the ΔS differences are picked from a set of size about 256^6.

(iii) : The size of the S sets (about 2^{16} pairs) is sufficient to test $(\delta_0, \delta_1, \delta_2, \delta_3)$, by comparing the sizes of S and ΔS, i.e. by counting how many collisions occur between the output xors for S pairs.

Proof (heuristic arguments) : In the first case, S is a subset of the $S_{\lambda_3, \alpha}$ considered in Proposition 2. Moreover, $\Delta S = \Delta S[14]$, because the two last rounds have no effect on the output xor of an S pair. The $\Delta S[14]$ elements are picked from a set of size about $|\Delta S_{\lambda_3,\alpha}[14]|/256^2$ because $\Delta S[14]$ contains only $\Delta S_{\lambda_3,\alpha}[14]$ elements $((\delta_{l_0}, \delta_{l_1}, \delta_{l_2}, \delta_{l_3}), (\delta_{r_0}, \delta_{r_1}, \delta_{r_2}, \delta_{r_3}))$ such that $\delta_{l_2} = 0$ and $\delta_{r_2} = 0$.
The first part of the claim now results from :
$|\Delta S_{\lambda_3,\alpha}[8]| = 1$ (by Proposition 2)
and
$|\Delta S_{\lambda_3,\alpha}[14]| \leq 128^6 |\Delta S_{\lambda_3,\alpha}[8]|$ (by Proposition 3).

The second part of the claim follows from the assumption that in other cases, the final xor values for pairs in the S set simply behave as random elements of the set of $((\delta_{l_0}, \delta_{l_1}, \delta_{l_2}, \delta_{l_3}), (\delta_{r_0}, \delta_{r_1}, \delta_{r_2}, \delta_{r_3}))$ differences such that $\delta_{l_2} = 0$ and $\delta_{r_2} = 0$.

The third part of the claim follows from the facts that since $2^{16} \gg (128^6/256^2)^{1/2}$ many collisions will occur in the first case, whereas few collisions will occur in the second case, since $2^{16} \ll (256^6)^{1/2}$.

Because of the difference between behaviours (i) and (ii), Substep 3.1 provides $c_1 \oplus c_3$, (as the value for which one δ_1 value behaves according to (i)), and also the $\delta_0\ \delta_1\ \delta_2\ \delta_3$ difference for the considered $\alpha \neq 0$ value. Substep 3.1 requires to encrypt about 2^{40} chosen plaintexts, and about 2^{40} operations.

Substep 3.2 : We are doing the same search as at Substep 3.1 for each of the 254 remaining $\alpha \neq 0$ values, but the $c_1 \oplus c_3$ byte has no longer to be exhaustively searched, since it has been determined at Substep 3.1.

Step 3 requires to encrypt about 2^{41} plaintext blocks, and the computational cost of Step 3 is about 2^{41} operations. After Step 3, the p_0, p_1, p_2 and p_3 permutations are entirely determined, up to the five unknown bytes c_3, $p_0[c_3]$, $p_1[c_3]$, $p_2[c_3]$, and $p_3[c_3]$, and the output of round 1 is known up to 8 constant bytes.

Step 4 is based on the results of Step 3.
We define four permutations $\overline{p}_0, \overline{p}_1, \overline{p}_2, \overline{p}_3$, by the relations :
$(\overline{p}_0, \overline{p}_1, \overline{p}_2, \overline{p}_3)[\lambda_3] = 0$;
$(\overline{p}_0, \overline{p}_1, \overline{p}_2, \overline{p}_3)[\lambda_3 \oplus \alpha] = (\delta_0; \delta_1, \delta_2, \delta_3)(\alpha)$ for $\alpha \neq 0$.
We thus define a $[\overline{p}_0, \overline{p}_1, \overline{p}_2, \overline{p}_3]$ S-box which is intended to be the basis for the construction of an equivalent representation of the first half of Khufu.
If we set $\overline{K1}_3 = 0$, the assumptions $[p_0, p_1, p_2, p_3] = [\overline{p}_0, \overline{p}_1, \overline{p}_2, \overline{p}_3]$ and $K1_3 = \overline{K1}_3$ provide the output of round 1 up to 8 unknown constant bytes for every input block.

The purpose of Step 4 is to gradually derive, one after the other, seven additional bytes $\overline{K2}_3, \overline{K1}_1, \overline{K2}_1, \overline{K1}_0, \overline{K2}_0, \overline{K1}_2, \overline{K2}_2$ such that :

- the assumptions $[p_0, p_1, p_2, p_3] = [\overline{p}_0, \overline{p}_1, \overline{p}_2, \overline{p}_3]$, $K1_3 = \overline{K1}_3$; $K2_3 = \overline{K2}_3$ provide the output of round 2 up to 8 unknown constant bytes for every input block;

⋮

- the assumptions $[p_0, p_1, p_2, p_3] = [\overline{p}_0, \overline{p}_1, \overline{p}_2, \overline{p}_3]$; $K1_3 = \overline{K1}_3$; $K2_3 = \overline{K2}_3$; $K1_1 = \overline{K1}_1$; $K2_1 = \overline{K2}_1$; $K1_0 = \overline{K1}_0$; $K2_0 = \overline{K2}_0$; $K1_2 = \overline{K1}_2$; $K2_2 = \overline{K2}_2$ provide the output of round 8 up to 8 unknown constant bytes for every input block.

Our estimate of the cost of the derivation of the above equivalent key bytes is at most 2^{40} plaintext blocks per key byte (2^{32} per test of an assumption on such a byte), using differential techniques similar to the one of Proposition 2 : introduction of a fixed difference in the S-box inputs of round r and compensation of the resulting difference for the S-box inputs of the subsequent rounds until round 8, with $r = 1$ in Proposition 2 and $r = 2$ to 8 here. Once the seven

above equivalent key bytes have been derived, a differential attack on the second half of Khufu (i.e. rounds 8 to 16 and the final xor with auxiliary key bits) can be mounted, at no substantial additional expense.

4 Aknowledgements

The attack presented in this paper is based on a preliminary investigation of Khufu in cooperation with Sean Murphy.
We also want to thank Thierry Baritaud for his help in the elaboration of the Latex version of this paper.

5 Conclusion

We have shown in some detail that first information concerning the secret S-box used in the first half of the scheme can be derived with about 2^{31} chosen plaintexts, and about 2^{31} operations (Step 1). Our estimate of the resources required for breaking the whole scheme is about 2^{43} chosen plaintexts and 2^{43} operations (Steps 2 to 4). However, further verifications (in particular computer experiments) are required to make sure that the figures announced in the description of Steps 3 and 4 are valid. Although the proposed attack is far from being realistic, because of the required amount of chosen plaintext, it suggests that the security of the 16-round Khufu might be too low for providing a suitable replacement to DES. In fairness to Merkle, it should be noticed however that some warnings concerning the choice of the 16-round version as compared with the 24-round and 32-round versions are already contained in [1], in particular the remark that the "safety factor" in Khufu with 16 rounds is less than that of DES.
We have found no similar attack for the 24-round version of Khufu.

References

1. Ralph Merkle, "Fast Software Encryption Functions", Advances in Cryptology - Crypto'90, Springer Verlag.
2. E. Biham and A. Shamir, "Differential Cryptanalysis of Snefru, Khafre, REDOC-II, LOKI, and Lucifer". Advances in Cryptology - Crypto'91, Springer Verlag.
3. E. Biham and A. Shamir, "Differential Cryptanalysis of the Data Encryption Standart". Springer Verlag, 1993, Chapter 7.