A Bottleneck Attack on Crypton

Marine Minier

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A bottleneck attack on Crypton

Marine Minier

N° 5324

Octobre 2004

Thème Cryptologie

ISSN 0249-6399 ISRN INRIA/RR--5324--FR+ENG

A bottleneck attack on Crypton

Marine Minier*

Thème Cryptologie — Cryptologie
Projet CODES

Rapport de recherche n° 5324 — Octobre 2004 — 14 pages

Résumé : Crypton est un candidat malchanceux de la compétition AES proposé par C. H. Lim en 1998. Nous présentons dans cet article deux attaques par goulet d'étranglement contre les deux versions de Crypton, Crypton v0.5 et Crypton v1.0. Ces attaques sont construites à partir d'une propriété particulière de trois tours de la fonction étage de Crypton liée à une dépendance restreinte entre des octets d'entrée et des octets de sortie. Cette propriété particulière permet de construire un distingueur sur 4 tours de la fonction étage. La complexité de l'attaque construite sur six étages à partir du distingueur se situe en dessous de la recherche exhaustive. Nous présentons également une amélioration de la recherche exhaustive dans le cas d'une version de Crypton à 7 tours.

Mots-clés : Cryptanalyses, chiffrement par blocs, Crypton, candidat AES

* Projet CODES - Bât 25, INRIA-Rocquencourt, Domaine de Voluceau, B.P. 105, F-78153 Le Chesnay Cedex, FRANCE

A Bottleneck Attack on Crypton

Abstract: Crypton is a 12-round blockcipher proposed as an AES candidate by C.H. Lim in 1998. In this paper, we present two bottleneck attacks on reduced round version of Crypton v0.5 and Crypton v1.0. Those cryptanalyses are built upon a four-round distinguisher based on a three-round property due to a restricted dependency of the one byte to one byte permutation transformation as made for the AES in [GM00].We present an attack on a six round version of Crypton. We also present a marginal speed up of the 128-bits key exhaustive search for a seven-round version of Crypton. This attack does not endanger the practical security offered by Crypton but shows an other example where the bottleneck property could be used with an S-box level composed of at least two S-boxes.

Key-words: Cryptanalysis, block ciphers, Crypton, AES candidat

1 Introduction

Crypton v0.5 [Li98] is a 12-round blockcipher which was submitted by C.H. Lim as one of the 15 candidates at the first Advanced Encryption Standard conference in August 1998. As many AES candidates, Crypton provides some provable resistance against linear and differential cryptanalyses.

Crypton v0.5 was not selected as an AES finalist due to the discovery of some weak keys by S. Vaudenay presented in [BGH+99]. A stochastic cryptanalysis described in [MG00] and based upon the statistical observations of [BGH+99] permits to mount an attack against an eight-round version of Crypton v0.5. A second version of Crypton, Crypton v1.0, was proposed in [Li99] with a modified S-box level and a modified key-schedule to make disappear the weak keys and to lower the number of differential and linear characteristics. In [HBR+99], C. D'Halluin et al. present a transposition of the Square attack to the 6-round Crypton v0.5 and v1.0 that permits to recover the entire 256-bits user key with a complexity of 2^{56} encryptions and 2^{32} plaintexts.

We present, in this paper, a bottleneck attack against a 6-round version of Crypton v0.5 and Crypton v1.0 for all the key lengths based on the same three-round property than the one used in the AES attack and presented in [GM00] and a marginal speed up of the 128-bits key exhaustive search for a seven-round version.

The rest of this paper is organized as follows: Section 2 briefly summaries the Crypton cipher. Section 3 presents the three inner round property and the four round distinguisher deduced which form the starting point of the attack. Section 4 presents the bottleneck attack against a 6 round version of Crypton. Section 5 describes the improvement of the128-bits key exhaustive search for a seven-round version and section 6 concludes this paper.

2 An Outline of Crypton

Crypton v0.5 and Crypton v1.0 encrypt 128-bit blocks with a key of length 128, 192 or 256 bits. The number r of rounds is 12. The algorithm consists of the encryption function itself and a keyschedule that derives $(r+1)$ 128-bit subkeys from the master key and an encryption function.

The encryption function consists of r rounds surrounded by an input transformation (which consists of an XOR between the plaintext block and the first subkey) and an output transformation ϕ that could be seen as a fixed modulo 2 linear mapping.

Let us represent a 128-bit block A by a 4×4 matrix of bytes:

$$A = \begin{pmatrix} a_{0,3} & a_{0,2} & a_{0,1} & a_{0,0} \\ a_{1,3} & a_{1,2} & a_{1,1} & a_{1,0} \\ a_{2,3} & a_{2,2} & a_{2,1} & a_{2,0} \\ a_{3,3} & a_{3,2} & a_{3,1} & a_{3,0} \end{pmatrix} \begin{matrix} A[0] \\ A[1] \\ A[2] \\ A[3] \end{matrix}$$

where each $a_{i,j}$ is a byte.
The round function ρ is composed by four basic transformations : a byte substitution γ, followed by a bit permutation π, a byte transposition τ and a subkey addition $\sigma_{k_e^i}$.

There are two main differences between Crypton v1.0 and Crypton v0.5 : the γ operation is modified and the keyschedule too. All the other operations are the same.

The four basic transformations are the following :

- γ (γ_o for odd round, γ_e for even round) uses two S-boxes S_0 and S_1 in the case of Crypton v0.5 and four S-boxes (S'_0, S'_1, $S'_2 = S_0^{'-1}$, $S'_3 = S_1^{'-1}$) deduced from one single S-box S in Crypton v1.0. For the Crypton v0.5 version, the γ_o transformation is given by the relation :

$$\begin{pmatrix} b_{0,3} & b_{0,2} & b_{0,1} & b_{0,0} \\ b_{1,3} & b_{1,2} & b_{1,1} & b_{1,0} \\ b_{2,3} & b_{2,2} & b_{2,1} & b_{2,0} \\ b_{3,3} & b_{3,2} & b_{3,1} & b_{3,0} \end{pmatrix} \xleftarrow{\gamma_o} \begin{pmatrix} S_1(a_{0,3}) & S_0(a_{0,2}) & S_1(a_{0,1}) & S_0(a_{0,0}) \\ S_0(a_{1,3}) & S_1(a_{1,2}) & S_0(a_{1,1}) & S_1(a_{1,0}) \\ S_1(a_{2,3}) & S_0(a_{2,2}) & S_1(a_{2,1}) & S_0(a_{2,0}) \\ S_0(a_{3,3}) & S_1(a_{3,2}) & S_0(a_{3,1}) & S_1(a_{3,0}) \end{pmatrix}$$

  To obtain γ_e from γ_o, one just needs to exchange S_0 and S_1. For Crypton v1.0, the γ_o and γ_e transformations are given by the relations :

$$B = \gamma_o(A) \Leftrightarrow b_{i,j} = S_{i+j \bmod 4}(a_{i,j})$$
$$B = \gamma_e(A) \Leftrightarrow b_{i,j} = S_{i+j+2 \bmod 4}(a_{i,j})$$

- π (π_o for odd rounds, π_e for even rounds) is a bit permutation. We only describe effects of π_o for odd rounds, effects of π_e are similar. π_o is given by :

$$\begin{aligned} T &= A[0] \oplus A[1] \oplus A[2] \oplus A[3], \\ B[0] &\leftarrow (A[0] \wedge MI_0) \oplus (A[1] \wedge MI_1) \oplus (A[2] \wedge MI_2) \oplus (A[3] \wedge MI_3) \oplus T, \\ B[1] &\leftarrow (A[0] \wedge MI_1) \oplus (A[1] \wedge MI_2) \oplus (A[2] \wedge MI_3) \oplus (A[3] \wedge MI_0) \oplus T, \\ B[2] &\leftarrow (A[0] \wedge MI_2) \oplus (A[1] \wedge MI_3) \oplus (A[2] \wedge MI_0) \oplus (A[3] \wedge MI_1) \oplus T, \\ B[3] &\leftarrow (A[0] \wedge MI_3) \oplus (A[1] \wedge MI_0) \oplus (A[2] \wedge MI_1) \oplus (A[3] \wedge MI_2) \oplus T, \end{aligned}$$

  where $MI_0 = \texttt{0xc0300c03}, MI_1 = \texttt{0x03c0300c}, MI_2 = \texttt{0x0c03c030}, MI_3 = \texttt{0x300c03c0}$ (in hexadecimal). We can notice that π is just a symmetry combined with a rotation for columns of 2-bit words.

  The π transformation could also be represented by 4 matrix "multiplications" in the vector space $GF(2)^8$ where $\oplus$ represents the internal addition and $\wedge$, the AND, the multiplication. For example, in the case of π_o, to compute the last column $(b_{3,0}, b_{2,0}, b_{1,0}, b_{0,0})^T$ of the output block B, we can multiply the last column $(a_{3,0}, a_{2,0}, a_{1,0}, a_{0,0},)^T$ of the input block A by the matrix :

$$\begin{pmatrix} m_0 & m_1 & m_2 & m_3 \\ m_1 & m_2 & m_3 & m_0 \\ m_2 & m_3 & m_0 & m_1 \\ m_3 & m_0 & m_1 & m_2 \end{pmatrix}$$

with $m_0 = \texttt{0xfc}$, $m_1 = \texttt{0xf3}$, $m_2 = \texttt{0xcf}$, $m_3 = \texttt{0x3f}$ in hexadecimal notation. We focus from now on this representation of π.

- The bytes transposition τ consists in exchanging all $a_{i,j}$ and $a_{j,i}$ pairs of bytes in the A matrix.
- The key addition $\sigma_{k_e^i}$ is an exclusive-or between the current block A and the 128-bit subkey k_e^i of the round i.

The final transformation ϕ is given by the relation : $\phi_e = \tau \circ \pi_e \circ \tau$ if the number of rounds is even and by $\phi_o = \tau \circ \pi_o \circ \tau$ if the number of rounds is odd.

So, in summary, the encryption function E_k of Crypton v0.5 and Crypton v1.0 is given by :

$$E_k = \phi_e \circ \rho_{e,k_e^r} \circ \rho_{o,k_e^{r-1}} \circ \cdots \circ \rho_{e,k_e^2} \circ \rho_{o,k_e^1} \circ \sigma_{k_e^0}$$

where k_e^i represents the subkey of the round i, $\rho_{e,k_e^i} = \sigma_{k_e^i} \circ \tau \circ \pi_e \circ \gamma_e$ represents the round function for an even round and $\rho_{o,k_e^i} = \sigma_{k_e^i} \circ \tau \circ \pi_o \circ \gamma_o$ represents the round function for an odd round.

This representation permits to show that the encryption and the decryption processes are strictly identical up to the keyschedule.

3 The Three-round Property and the Four-round Distinguisher Deduced

As mentioned in [HBR+99], the structure of Crypton is very similar to the Square-like ciphers, so it is not really surprising that we could apply on Crypton the bottleneck attack even if the linear part does not verify the MDS property described in [DR02] (for example, if we transform the 128-bit word $\texttt{0x00}\cdots\texttt{001}$ with the linear part of Crypton, only three bytes are modified in the output instead of four for an MDS linear part). Nevertheless, we could say that the linear part of Crypton verifies the MDS property for the integral cryptanalysis (i.e. if an active byte takes all possible values between 0 and 255 and all the other bytes are taken equal to a constant, then after one round, it influences exactly four bytes) that is a necessary (but not sufficient) condition for a bottleneck attack.

We are going to see in this section the three inner rounds property used and the four rounds distinguisher deduced based on a restricted dependency between an input active byte (that takes all possible values between 0 and 255) and a corresponding output byte.

3.1 The Three-round Property

We only study here the case where the first round is an odd round but the same result holds if the first round is an even round due to the very symmetric structure of π_o and π_e. This property is also valid for the both versions of Crypton (Crypton v0.5 and Crypton v1.0) due to the fact that the S-box level has no influence on the partial bijection of some

involved functions. So, from now, we denote by $S_i, i = 0..3$, all the S-boxes crossed in the three inner rounds.

We denote by B, C, D and E the different intermediate input/output states of three consecutive inner rounds as noticed in figure 1.

We focus from now on an input block B with its three lowest rows fixed. The highest row, marked on figure 1, is composed by one active byte y which takes all possible values between 0 and 255 and by a triplet b equal to (b_0, b_1, b_2) of constant bytes which will represent a parameter. More formally, we have $B_{0,3} = b_0$, $B_{0,2} = b_1$, $B_{0,1} = c_1$, $B_{0,0} = b_2$.

In the same way, we also use the following notations for some particular bytes marked in figure 1. So, we denote $C_{0,3} = c_0$, $C_{0,2} = c_1$, $C_{0,1} = c_2$, $C_{0,0} = c_3$, $D_{3,0} = d_0$, $D_{2,0} = d_1$, $D_{1,0} = d_2$, $D_{0,0} = d_3$ and $E_{0,0} = s$.

We also note

$$(\pi_o(\gamma_o(B)))_{3,0} = y'_0 \quad (\pi_o(\gamma_o(B)))_{2,0} = y'_1 \quad (\pi_o(\gamma_o(B)))_{1,0} = y'_2$$
$$(\pi_o(\gamma_o(B)))_{0,0} = y'_3 \quad (\pi_e(\gamma_e(D)))_{0,3} = c'_0 \quad (\pi_e(\gamma_e(D)))_{0,2} = c'_1$$
$$(\pi_e(\gamma_e(D)))_{0,1} = c'_2 \quad (\pi_e(\gamma_e(D)))_{0,0} = c'_3 \quad \text{and} \quad (\pi_o(\gamma_o(D)))_{0,0} = d_4$$

some intermediate bytes also marked on figure 1.

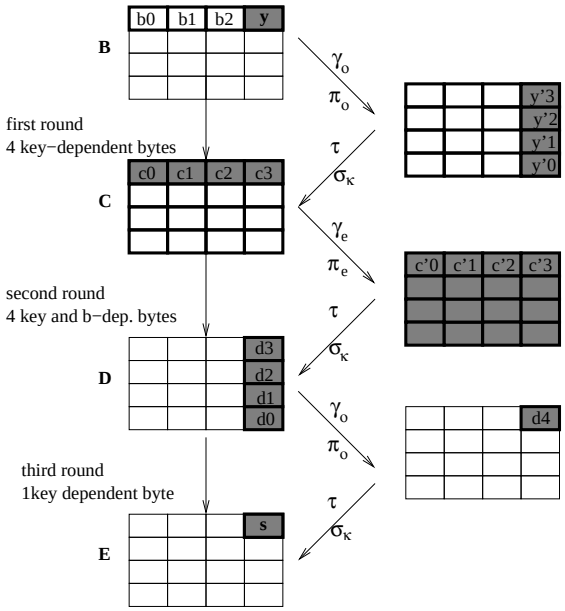

FIG. 1: *The Three Inner Rounds Property*

So, let us analyze how the C, D and E particular bytes c_0 to c_3, d_0 to d_3, and s can be seen as b-dependent and key-dependent functions of the y input byte.

- After the first round, the $y \rightarrow c_0^b[y]$ one to one function is independent from the value of the b triplet and is entirely determined by one key byte, due to the effect of the π_o and of the τ operations. Indeed, all the y'_i bytes are entirely determined by the y value. The same property holds for c_1, c_2 and c_3. So, the quartet of bytes (c_0, c_1, c_2, c_3) is a function of the y values entirely determined by four key-dependent bytes. More formally, if we denote by $k_e^{1,0,i}$ for $i = 0..3$ the 4 key-dependent constants of the round subkey, we have:

$$\begin{aligned} c_i &= y'_i \oplus k_e^{1,0,i} = \pi_o(\gamma_o(B))_{i,0} \oplus k_e^{1,0,i} \ , \ i = 0..3 \\ c_i &= (m_{3-i} \wedge S_{3-i}(y)) \oplus \delta_i \oplus k_e^{1,0,i} \ , \ i = 0..3 \end{aligned}$$

  where δ_i, $i = 0..3$, are four constants depending on the fixed values of the three lowest rows.

- After the second round, each of the four bytes $d_i[y]$, $i = 0..3$ is a one to one function of the corresponding $c_i[y]$ byte entirely determined by one single unknown byte that is entirely determined by b and the key. The quartet of bytes (d_0, d_1, d_2, d_3) of D marked on figure 1 is a function of (c_0, c_1, c_2, c_3) entirely determined by four key-dependent and b-dependent bytes. More formally, if we denote by $k_e^{2,3-i,0}$ for $i = 0..3$ the four key-bytes implied by the subkey addition, we have:

$$\begin{aligned} d_i &= c'_i \oplus k_e^{2,3-i,0} = \pi_e(\gamma_e(C))_{0,3-i} \oplus k_e^{2,3-i,0} \ , \ i = 0..3 \\ d_i &= \left(m_{4-i \bmod 4} \wedge S_{1-i \bmod 4}(c_i)\right) \oplus \delta'_i(b) \oplus k_e^{2,3-i,0} \ , \ i = 0..3 \\ \text{with } \delta'_i(b) &= \bigoplus_{j=1}^{3} \left(m_{j-i \bmod 4} \wedge S_{i+j+2 \bmod 4}(C_{j,3-i})\right) \ , \ i = 0..3. \end{aligned}$$

  So, in summary The quartet of bytes (d_0, d_1, d_2, d_3) is a one to one function of (c_0, c_1, c_2, c_3) entirely determined by the four key-dependent and b-dependent bytes $\delta'_i(b) \oplus k_e^{2,3-i,0}$, $i = 0..3$.

- After the third round, the s byte marked on figure 1 can be expressed as a function of the (d_0, d_1, d_2, d_3) quartet of bytes entirely determined by one key-dependent byte depending on the subkey of the round. If we denote by $k_e^{3,0,0}$ the key byte used, we have:

$$s = d_4 \oplus k_e^{3,0,0} = \bigoplus_{i=0}^{3} \left(m_{3-i \bmod 4} \wedge d_i\right) \oplus k_e^{3,0,0}$$

In summary, the s byte depends on only 5 key-dependent bytes and 4 b-dependent bytes. The partial function $s^b[y]$ is entirely determined by a reduced number of unknown bytes.

We can exploit this restricted dependency by constructing collisions on all the y values for distinct values of the b triplet. In other words :

Property 1 *There exists b' and b'' two triplets of constants such as for all y values between 0 and 255, we have : $s^{b'}[y] = s^{b''}[y]$. In this case, we say that we have a collision.*

In fact, the number of obtained collisions is 256, one for each y value.

This property was verified by computer experiments and works for Crypton v0.5 and Crypton v1.0 and also if the first round involved in the property is odd or even. This fact seems to be natural since only the structure of the π transformation is really used and all the equations given in this section stay true if the first round is even up to the m_i values.

Under the heuristic assumption that the unknown constants depending on the key and on the b triplet behave as random functions, then, by the birthday paradox, if we take a $\mathbb{B}$ set of 2^{16} b triplets, the probability to obtain a collision is non negligible.

This property can be extended to mount an efficient four-rounds distinguisher by adding a fourth round at the bottom of the three previous rounds.

3.2 The Four-round Distinguisher

We consider the deciphering of the fourth round in the following way :

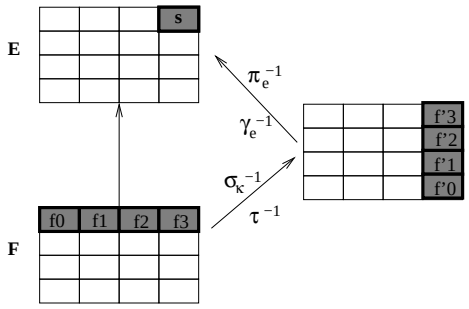

FIG. 2: *The Extension to a Fourth Round*

We denote by F the output block after the fourth round and we denote by (f_0, f_1, f_2, f_3) the first row of F marked on the figure 2. We have the following relations : $f'_0 = f_0 \oplus k_e^{4,0,3}$, $f'_1 = f_1 \oplus k_e^{4,0,2}$, $f'_2 = f_2 \oplus k_e^{4,0,1}$, $f'_3 = f_3 \oplus k_e^{4,0,0}$. Since we have $\pi_e^{-1} = \pi_e$ and $\pi_o^{-1} = \pi_o$, we can express the byte s as

$$\begin{aligned}
s &= S^{-1}\left((f'_3 \wedge m_1) \oplus (f'_2 \wedge m_2) \oplus (f'_1 \wedge m_3) \oplus (f'_0 \wedge m_0)\right) \\
s &= S^{-1}\left((f_3 \wedge m_1) \oplus (f_2 \wedge m_2) \oplus (f_1 \wedge m_3) \oplus (f_0 \wedge m_0)\right) \\
&\quad \oplus \left((k_e^{4,0,3} \wedge m_0) \oplus (k_e^{4,0,2} \wedge m_3) \oplus (k_e^{4,0,1} \wedge m_2) \oplus (k_e^{4,0,0} \wedge m_1)\right) \\
s &= S^{-1}\left((f_3 \wedge m_1) \oplus (f_2 \wedge m_2) \oplus (f_1 \wedge m_3) \oplus (f_0 \wedge m_0) \oplus \delta\right)
\end{aligned}$$

where S represents the good Crypton S-box and δ a constant depending on the subkey of the fourth round. We have the following property :

Property 2 *There exists a collision between $s^{b'}[y]$ and $s^{b''}[y]$ if and only if for all y values between 0 and 255, we have :*

$$(f_0^{b'}[y] \wedge m_0) \oplus (f_1^{b'}[y] \wedge m_3) \oplus (f_2^{b'}[y] \wedge m_2) \oplus (f_3^{b'}[y] \wedge m_1)$$
$$=$$
$$(f_0^{b''}[y] \wedge m_0) \oplus (f_1^{b''}[y] \wedge m_3) \oplus (f_2^{b''}[y] \wedge m_2) \oplus (f_3^{b''}[y] \wedge m_1)$$

To simplify the notations we denote $(f_0^b[y] \wedge m_0) \oplus (f_1^b[y] \wedge m_3) \oplus (f_2^b[y] \wedge m_2) \oplus (f_3^b[y] \wedge m_1)$ by $\mathcal{F}^b[y]$. $\mathcal{F}^b[y]$ is a function of y entirely determined by 6 unknown bytes depending on the key and by 4 additional unknown bytes depending on both the key and the b values.

We test the following four inner round distinguisher on a limited number of y values, a set Λ of 16 values is sufficient, the number of false alarms being negligible in this case.

- Select a $\mathbb{B}$ set of about 2^{16} b triplet values and a subset of $\{0 \cdots 255\}$, say for instance a Λ subset of 16 y values.
- For each b triplet value, compute the values
$$L_b = \left((f_0^b \wedge m_0) \oplus (f_1^b \wedge m_3) \oplus (f_2^b \wedge m_2) \oplus (f_3^b \wedge m_1)\right)_{y \in \Lambda}.$$
- Check whether two of the above lists, $L_{b'}$ and $L_{b''}$ are equal.

We claim that the computations made at the second step of this distinguisher ($2^7 = 16 \cdot 8$ elementary operations) represent substantially less than one single Crypton execution.

This four-round distinguisher requires about 2^{20} chosen inputs B, and since the collision detection computations (based on the analysis of the corresponding F values) require less operations than the 2^{20} 4-inner rounds computations, the complexity of the distinguisher is less than 2^{20} Crypton encryptions.

4 The Bottleneck Attack on a Six-round Version of Crypton

In this section, we show how we can extend the four inner rounds distinguisher of the previous section to mount a six-round attack for all the key lengths by adding a first round at the beginning and a sixth round at the end followed by the ϕ transformation.

if we denote by A the plaintext block and by G the corresponding ciphertext, the four previous rounds are surrounded at the top by one initial round A $\rightarrow$ B, composed by an initial key addition followed by one round and at the bottom by one final round: F $\rightarrow$ G. We do not take here into account the ϕ transformation that could be considered as a bit permutation.

This attack method is a combination between the four-round distinguisher presented in section 4 and an exhaustive search of some keybytes or combination of keybytes of the initial and the final rounds.

4.1 Extension at the Beginning

We could extend the previous distinguisher by one round at the beginning using the same method than the one proposed in [HBR+99] and also used in the Square Attack against the AES [DR98].

The main idea used here is that if, in the initial key addition using k_e^0, the 4 key bytes (that we denote by $k_{ini} = (k_e^{0,0,0}, k_e^{0,1,0}, k_e^{0,2,0}, k_e^{0,3,0})$), added with the four bytes $(a_{0,0}, a_{1,0}, a_{2,0}, a_{3,0})$ of the plaintext A, are known then it is possible to partition the 2^{32} plaintexts into 2^{24} subsets of 2^8 plaintexts values satisfying the conditions of section 3.1.

So, if we encrypt all the 2^{32} possible plaintexts for all the possible values of the $(a_{0,0}, a_{1,0}, a_{2,0}, a_{3,0})$ quartet (the other 12 bytes being taken equal to a constant), we could, according to the value of k_{ini}, partition the 2^{32} plaintexts into 2^{24} subsets of 2^8 plaintexts according the values of y (which are known up to an unknown constant linked with the first round key byte $k_e^{1,0,0}$). Those subsets are such that the y byte takes all possible values between 0 and 255 and the $b = (b_0, b_1, b_2)$ triplet is composed of three constant values, different and unique for each of the 2^{24} subsets, the 12 other B bytes are constant and all those constant values are the same for all subsets.

Those 2^{32} plaintexts give the corresponding 2^{32} ciphertexts G.

4.2 Extension at the End

Each of the f_0, f_1, f_2, f_3 bytes can be expressed as a function of four bytes of the G ciphertext and one unknown key bytes, linear combination of four bytes of the last round subkey up to the ϕ transformation.

Indeed, if we do not take into account the ϕ transformation, we have, for i from 0 to 3 :

$$\begin{aligned} f_i &= S_i^{-1}\left(\bigoplus_{j=0}^{3}\left(G_{j,3-i} \wedge m_{j+3-i \bmod 4}\right) \oplus \Delta_i\right) \\ \text{with } \Delta_i &= \bigoplus_{j=0}^{3}\left(k_e^{6,j,i} \wedge m_{j+3-i \bmod 4}\right) \end{aligned}$$

where S_i represents the appropriated S-box involved.

So, we could make an exhaustive search on the four Δ_i bytes to compute the f_i values from the ciphertext up to ϕ and to test the four-round distinguisher.

4.3 Outline of the Attack for the Different Key Lengths

We perform an exhaustive search of the k_{ini} and Δ_i values in the following way :

First Step :
Cipher the 2^{32} chosen plaintexts for all possible values
of the quartet $(a_{0,0}, a_{1,0}, a_{2,0}, a_{3,0})$.
Second step :
For k_{ini} from (0,0,0,0) to (255,255,255,255) do
  Partition the $(256)^4$ chosen plaintexts
  into $(256)^3$ $\mathbb{B}^b$ sets according the value of the triplet b
  Choose into those $(256)^3$ $\mathbb{B}^b$ sets 2^{16} values of b
  For $(\Delta_0, \cdots, \Delta_3)$ from $(0, \cdots, 0)$ to $(255, \cdots, 255)$ do
    For each value of the b triplet do
      Compute the values of $L_b = \left(\mathcal{F}^b[y]\right)_{y \in \Lambda}$ from the ciphertexts
      Check whether two of the above lists $L_{b'}$ and $L_{b''}$ are equal
      If yes, verify the same computation for all the y values
        If equality for all y values, return $(k_{ini}, \Delta_0, \cdots, \Delta_3)$
      Else continue
      End If
    End For
  End For
End For

Since the above procedure tests if there exists some collisions inside a random set of 256^2 of the 256^4 possible $\mathcal{F}^b[y]$ functions, the probability of the procedure to result in a collision, and thus to provide k_{ini}, Δ_0, Δ_1, Δ_2 and Δ_3 is high (say about $1/2$). In other words, the success probability of the attack is about $1/2$.

The first step could be made independently and requires 2^{32} chosen plaintexts and 2^{32} Crypton executions.

The complexity of the second step is about 2^{84} operations (2^{64} for the key exhaustive search and 2^{20} for the distinguisher itself) less expensive than Crypton executions. Its probability of success is about $1/2$. This attack provides 4 bytes of information on the first subkey and four other bytes of information on the sixth subkey.

5 A Speed up of the Key Exhaustive Search on 7 rounds for a 128 bits Key

It is also possible to extend this cryptanalysis to a speed up of the 128-bits k user key exhaustive search for a seven-round version of Crypton for the different key lengths as made for the AES in [GM00] by adding a seventh round at the bottom G $\rightarrow$ H and by completely computing k_e^7.

As made in [GM00], we know that the s byte value is entirely determined by 12 bytes depending on the round subkeys. More precisely, the value of the quartet of bytes $(d_0, \cdots, d_3)$ is entirely determined by the $(c_0, \cdots, c_3)$ value and by 12 additional bytes $\Psi(k)$ depending on the round subkeys.

So, we could use the following algorithm with a large amount of pre-computations to improve the key exhaustive search :

Pre-computations 1 :
For all possible values of $\Psi(k)$ do
  Compute the colliding values b' and b'' by testing a subset $\mathbb{B}^b$
  of about $(256)^2$ b values (using the three-round property)
  Store b' and b'' in a table T1$[\Psi(k)]$
First Step :
Cipher the 2^{32} chosen plaintexts for all possible values
of the quartet $(a_{0,0}, a_{1,0}, a_{2,0}, a_{3,0})$.
Pre-computations 2 :
For all k_{ini} values and for all $(256)^2$ b possible values do
  Decipher the first round
  Compute the plaintexts $\mathrm{A}^b[y]$ corresponding to the 16 y values in Λ
  Store in a table T2$[k_{ini}, b]$ the 16 corresponding ciphertexts $(\mathrm{H}^b[y])_{y\in\Lambda}$
Second step : key exhaustive search
For all the k user key values do
  Compute k_{ini}, $\Psi(k)$, k_e^7, Δ_0, $\cdots$, Δ_3
  Find the colliding values b' and b'' in T1$[\Psi(k)]$
  Find in T2$[k_{ini}, b']$ and in T2$[k_{ini}, b'']$, $(\mathrm{H}^{b'}[y])_{y\in\Lambda}$ and $(\mathrm{H}^{b''}[y])_{y\in\Lambda}$
  For the k_e^7, Δ_0, $\cdots$, Δ_3 values found do
    Decipher $(\mathrm{H}^{b'}[y])$ and $(\mathrm{H}^{b''}[y])$ for the first y value
    Test if $\mathcal{F}^{b'}[y] = \mathcal{F}^{b''}[y]$ for the first y value
    If equality then test the second y value and so on...
    If equality for all the y values in Λ then
      Test the k value for few plaintexts and ciphertexts.
    End If
End For

In this improvement of the key exhaustive search, the first step of pre-computations could fail due the fact that we test only $(256)^2$ b values, so we could say that the probability of success of this algorithm is about $1/2$. The complexity of this first pre-computation phase is about 2^{116} Crypton executions and the memory required is about 2^{98} bytes. The memory required in the second pre-computation step is about 2^{60} bytes.

The computations made in the second step are for each key value less expensive than a Crypton encryption (a complete call to the keyschedule and two partial decryptions). So, the total complexity of the algorithm is less than 2^{128} Crypton executions and requires 2^{99} bytes of memory.

6 Conclusion

We sum up in the following tables all the cryptanalytic results concerning Crypton v0.5 and Crypton v1.0 :

Crypton v0.5

Attack	Type	Nb rounds	Plaintexts	Time
[HBR+99]	Square Att.	6	2^{32}	2^{56}
[MG00]	Stochastic Att.	8	2^{112}	2^{113}
This paper	Bottleneck Att.	6	2^{32}	2^{84}
This paper	Bottleneck Att.	7	2^{32}	$< 2^{128}$

Crypton v1.0

Attack	Type	Nb rounds	Plaintexts	Time
[HBR+99]	Square Att.	6	2^{32}	2^{56}
This paper	Bottleneck Att.	6	2^{32}	2^{84}
This paper	Bottleneck Att.	7	2^{32}	$< 2^{128}$

We have described one cryptanalysis on a six-round version of the block cipher Crypton, faster than an exhaustive search and an improvement of the key exhaustive search on a seven-round version of Crypton. The attacks presented here do not threaten the security of Crypton in a full version but put the stress on a weakness of the linear part. This is an other example where the bottleneck attack could be used with the same properties than the one described in [GM00].

Références

[BGH+99] O. Baudron, H. Gilbert, L. Granboulan, H. Handschuh, A. Joux, P. Nguyen, F. Noilhan, D. Pointcheval, T. Pornin, G. Poupard, J. Stern, S. Vaudenay, "Report on the AES Candidates", *The Second Advanced Encryption Standard Candidate Conference,* N.I.S.T., 1999.

[DR98] J. Daemen, V. Rijmen, "AES Proposal : Rijndael", *The First Advanced Encryption Standard Candidate Conference,* N.I.S.T., 1998.

[GM00] H. Gilbert, M. Minier, "A Collision Attack on 7 rounds of Rijndael". In *The Third Advanced Encryption Standard Candidate Conference.* N.I.S.T., 2000.

[HBR+99] C. D'Halluin, G. Bijnens, V. Rijmen, B. Preneel, "Attack on Six Rounds of Crypton". In *Fast Software Encryption - FSE'99* , p. 46, LNCS 1267, Springer Verlag, Rome, Italy, March 1999.

[Li98] C.H. Lim, "Crypton : A New 128-bit Block Cipher", *The First Advanced Encryption Standard Candidate Conference,* N.I.S.T., 1998.

[Li99] C.H. Lim, "A Revised Version of Crypton : Crypton V1.0". In *Fast Software Encryption - FSE'99* , p. 31, LNCS 1267, Springer Verlag, Rome, Italy, March 1999.

[MG00] M. Minier, H, Gilbert, "Stochastic Cryptanalysis of Crypton". In *Fast Software Encryption - FSE'00* , p. 121, LNCS 1978, Springer Verlag, New York, April 2000.

Table des matières

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399