# Linear Analysis of reduced-round CAST-128 and CAST-256

**Jorge Nakahara Jr**[1], **Mads Rasmussen**[2]

[1] UNISANTOS, Brazil

jorge_nakahara@yahoo.com.br

[2] LSI-TEC, PKI Certification department

mads@lsitec.org.br

***Abstract.*** *This paper*[1] *describes a linear analysis of reduced-round versions of the CAST-128 and CAST-256 block ciphers. CAST-256 was a former candidate to the AES Development Process. Both ciphers use the same nonlinear components (fixed* $8\times32$*-bit S-boxes, key-dependent bit-rotation, modular addition and subtraction on 32-bit words) and a Feistel Network structure. We exploit the fact that the S-boxes are non-surjective mappings to construct iterative linear distinguishers for both ciphers. As far as we are aware of, this paper describes the first known-plaintext analysis of reduced-round variants of these ciphers.*

Keywords: CAST-128, CAST-256, linear cryptanalysis.

## 1. Introduction

This paper describes a linear analysis of reduced-round versions of the CAST-128 and CAST-256 block ciphers. There are many different versions of CAST ciphers reported in the literature [2, 4, 5, 15, 23, 26]. Consequently, attacks such as non-surjective, differential and the related-key techniques have already been applied to these other CAST variants. Nonetheless, no known-plaintext attacks have been applied to CAST-128 and CAST-256, in particular. Even though our attacks do not threaten any full-round cipher version, they help justify the selected number of rounds, and may serve as a first step towards more powerful attacks.

CAST-128 is block cipher designed by C. Adams and S. Tavares in 1996 [4]. CAST-128 operates on 64-bit text blocks under a key of variable size, 40 bits up to 128 bits, in increments of 8 bits. This cipher has a Feistel Network structure (like the DES [24]) with 12 or 16 rounds [2]. In real applications, CAST-128 is part of the suite of symmetric ciphers used by GnuPG [13, 31], under the name CAST-5.

The Feistel structure of CAST-128 operates on a 64-bit plaintext block $M_0 = (L_0, R_0)$, where both $L_0$ and $R_0$ are 32-bit strings.

- key schedule: compute 16 pairs of subkeys $(K_{mi}, K_{ri})$ from the user key $K$, with one pair of subkeys per round. A 32-bit key-dependent value $K_{mi}$ is used as a "masking" key and a 5-bit quantity $K_{ri}$ is used as a "rotation" key of the $i$-th round.
- for $1 \leq i \leq 16$ compute $L_i$ and $R_i$ as follows: $L_i = R_{i-1}$ and $R_i = L_{i-1} \oplus f_i(R_{i-1}, K_{mi}, K_{ri})$, where $f_i$ is the round function ($f_i$ may be of Type 1, Type 2 or Type 3, depending on $i$), described later on.

[1] First author funded by FAPESP under contract 2005/02102-9.

- the ciphertext is $(R_{16}, L_{16})$.

From [2], there are three different round functions in CAST-128. These round functions, denoted $f_i$, operate on the data input represented by $I = I_a|I_b|I_c|I_d$ where $I_a$ until $I_d$ are the most significant byte through the least significant byte of $I$, respectively. The symbol | denotes bitstring concatenation. The symbols "+" and "-" denote addition and subtraction modulo $2^{32}$, $\oplus$ is bitwise XOR, and $\lll$ denotes the circular left-shift operation.

Type 1: $I = ((K_{mi} + X) \lll K_{ri})$

$$f_1 = ((S_1[I_a] \oplus S_2[I_b]) - S_3[I_c]) + S_4[I_d]$$

Type 2: $I = ((K_{mi} \oplus X) \lll K_{ri})$

$$f_2 = ((S_1[I_a] - S_2[I_b]) + S_3[I_c]) \oplus S_4[I_d]$$

Type 3: $I = ((K_{mi} - X) \lll K_{ri})$

$$f_3 = ((S_1[I_a] + S_2[I_b]) \oplus S_3[I_c]) - S_4[I_d]$$

Rounds 1, 4, 7, 10, 13 and 16 use $f_1$ (Type 1 function), while rounds 2, 5, 8, 11 and 14 use $f_2$ (Type 2 function), and finally, rounds 3, 6, 9, 12 and 15 use $f_3$ (Type 3 function).

CAST-128 uses eight (fixed) substitution boxes: $S_1$, $S_2$, $S_3$, $S_4$ for the round function, and $S_5$, $S_6$, $S_7$, $S_8$ for the key schedule. Although these S-boxes require a total of eight KBytes of storage, only four KBytes are required during actual encryption and decryption since subkey generation is typically done prior to any data input.

Since our attacks do not depend on the key schedule, we omit its description. We refer to [2, 4] for further details.

## 2. CAST-256

The CAST-256 block cipher is a former candidate to the AES Development Process [1, 3]. Even though CAST-256 was not among the finalists to the AES Process, its analysis may help understand the design rationale of other ciphers from the CAST family.

CAST-256 operates on 128-bit text blocks under keys of 128, 192 or 256 bits. CAST-256 has a Generalized Feistel Network structure [28] and iterates 48 rounds for all key sizes.

Let a plaintext block be denoted $P = (A, B, C, D)$, where each of $A$, $B$, $C$ and $D$ is a 32-bit string. Let a quad-round be defined as the following four rounds, in order:

$$C = C \oplus f_1(D, K_{r0_i}, K_{m0_i});$$

$$B = B \oplus f_2(C, K_{r1_i}, K_{m1_i});$$

$$A = A \oplus f_3(B, K_{r2_i}, K_{m2_i});$$

$$D = D \oplus f_1(A, K_{r3_i}, K_{m3_i});$$

where $K_{rj_i}$ and $K_{mj_i}$ are generated by the key schedule algorithm of CAST-256 (further details of the key schedule algorithm can be found in [3]). The inverse quad-round is defined as

$D = D \oplus f_1(A, K_{r3_i}, K_{m3_i});$

$A = A \oplus f_3(B, K_{r2_i}, K_{m2_i});$

$B = B \oplus f_2(C, K_{r1_i}, K_{m1_i});$

$C = C \oplus f_1(D, K_{r0_i}, K_{m0_i});$

Notice that since exclusive-or is an involution, the $f_i$ mappings need not be invertible for the decryption operation.

The encryption of a 128-bit plaintext block in CAST-256 consists of six quad-rounds followed by six inverse quad-rounds (or 48 single rounds).

Previous analysis of ciphers of the CAST type include [14, 15, 23] but their analyses do not apply to either CAST-128 nor CAST-256. In [30], Sung *et al.* described a related-cipher attack on 4-round CAST-128 requiring $2^{17}$ chosen plaintexts and adaptively-chosen ciphertexts (CPACC), and $2^{40}$ encryptions.

In [26], Rijmen *et al.* describe non-surjective attacks on 6- and 8-round versions of (old) CAST designs denoted $\text{CAST}_{32}$, in which the round function did not mix exclusive-or with modular addition. We have found that the round functions $f_1$, $f_2$ and $f_3$ above are non-surjective, but the fraction of output values of each one of them are $1 - 1580054165/2^{32} \approx 0.63211$; $1 - 1580108708/2^{32} \approx 0.6321$ and $1 - 1580028865/2^{32} \approx 0.63212$, respectively. All of these values are close to $1 - 1/e$, which is the expected number of output values for a random function [26]. So, non-surjective attacks do not seem effective against CAST-256 and CAST-128.

In [5], Adams *et al.* provided some reasoning to justify the security of the full CAST-256 against conventional differential and linear analysis. But, they do not discuss the security of reduced-round variants of CAST-256.

In [6], Biham claimed an attack on 20-round CAST-256 by the impossible differential technique, but no details were provided (see Sect. 6.).

## 3. Linear Cryptanalysis

The linear cryptanalysis (LC) technique was developed by Matsui and applied successfully against Feistel ciphers such as the DES [20] and FEAL [22]. This technique is one of the most general known attacks on block ciphers, and has become a benchmark technique for evaluating the security of any new modern cipher (AES [12], LOKI97 [9, 17], RC6 [27]). LC is a known-plaintext (KP) attack, but it has already been used in other settings such as chosen-plaintext [18] and ciphertext-only [19], which makes it quite attractive in a real-life setting.

The fundamental tool of a linear attack is a linear distinguisher which consists of a linear equation involving bits of plaintext, ciphertext and key, holding with non-uniform probability. This discrepancy between the probability of a linear relation of a cipher and that of a random behavior is called bias. Usually, linear relations are derived for each individual component in a cipher. Further, the linear relations are combined, leading to linear approximations of larger structures, until reaching multiple rounds. Intuitively, linear approximations are performed preferably on the nonlinear components, such as S-boxes. Let an S-box $S : \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^m$, and two bit strings, $\Gamma X \in \mathbb{Z}_2^n$ and $\Gamma Y \in \mathbb{Z}_2^m$, known as bit masks. The linear relation involving the input bits of $S$, designated by $\Gamma X$, and

its output bits, designated by $\Gamma Y$, is $X \cdot \Gamma X \oplus S[X] \cdot \Gamma Y = 0$. The probability that this relation holds is

$$P_{\Gamma X, \Gamma Y} = \frac{\#\{X \in \mathbb{Z}_2^n | X \cdot \Gamma X = S[X] \cdot \Gamma Y\}}{\#\{X \in \mathbb{Z}_2^n\}} . \quad (1)$$

If $\Gamma X = \Gamma Y = 0$, then the bitmask is called trivial; otherwise, it is called non-trivial. The bias of this linear relation is $|P_{\Gamma X, \Gamma Y} - 1/2|$. An exhaustive list containing all input and output bit masks of an S-box $S$ is called the Linear Approximation Table (LAT) of $S$ [20]. The LAT allows one to identify the most promising (non-trivial) linear relations for $S$, namely the ones with highest bias.

The bias of the combination of two linear approximations is derived using Matsui's Piling-Up Lemma [20]. We will employ this lemma even though it is not always strictly correct [8, 29]. The Piling-Up Lemma assumes all round subkeys are independent, in order to calculate the combined bias of independent linear relations. The round subkeys in CAST, though, are not independent but generated via a key schedule algorithm. Nonetheless, we assume the approximation is reasonably good, as already assumed in previous linear attacks on the DES cipher [20].

But, one cannot combine an arbitrary number of linear relations. An obvious restriction is to limit the attack effort to less than that of an exhaustive key search. The key size for CAST-128 is at least $40$ bits, and for CAST-256 at least $128$ bits. Moreover, for both CAST-128 and CAST-256, we have looked for attacks that did not require more text than the full codebook ($2^{64}$ and $2^{128}$ text blocks, respectively). Otherwise, an attacker could collect the codebook and use it decrypt (and forge) cryptograms without knowing the key (and while the key is not changed).

## 4. Linear Analysis of CAST-128

The four S-boxes of CAST-128 have dimension $8 \times 32$ bits, and thus, are non-surjective. Consequently, we looked for linear relations for these S-boxes with the form $0 \overset{S-box}{\rightarrow} \Gamma$, where $0$ stands for a zero 8-bit mask, and $\Gamma$ stands for a nonzero (non-trivial) 32-bit mask. This notation means that the exclusive-or of no input bits to the S-boxes causes an exclusive-or of output bits selected by $\Gamma$ (with a nonzero bias). It means that the exclusive-or of only output bits is zero. Naturally, this linear relation only makes sense if the associated probability is away from 1/2.

Due to the modular addition and subtraction operations in the round function of CAST-128, we have looked for bitmasks $\Gamma$ with nonzero bits preferably in the least significant bit positions, to avoid decreasing the bias due to carry and borrow bits in the modular addition and subtraction operations. The best choice is $\Gamma = 1$ (Fig. 1).

Thus, the linear relations we chose for the round functions $F$ have either the form $\texttt{00000000}_\texttt{x} \overset{F}{\rightarrow} \texttt{00000000}_\texttt{x}$ or $\texttt{00000000}_\texttt{x} \overset{F}{\rightarrow} \texttt{00000001}_\texttt{x}$, where the subscripts $_\texttt{x}$ indicate hexadecimal value. The rationale is to maximize the bias, and to minimize the number of active S-boxes in building linear relations across the round function (a strategy already employed against the DES cipher [21]). Nonetheless, due to the construction of the $F$ function, either all four S-boxes are active or none is. An active S-box is effectively used in a linear approximation, in the sense that either the input or the output mask is nonzero (non-trivial).

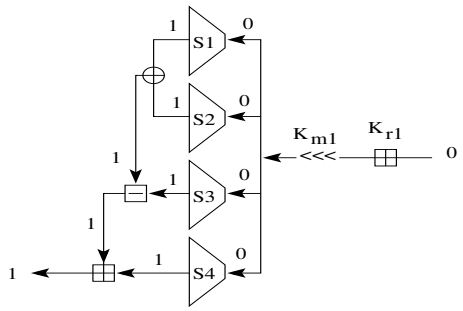


**Figure 1. Bit masks showing the propagation of a linear relation across the round function $F$ of CAST-128.**

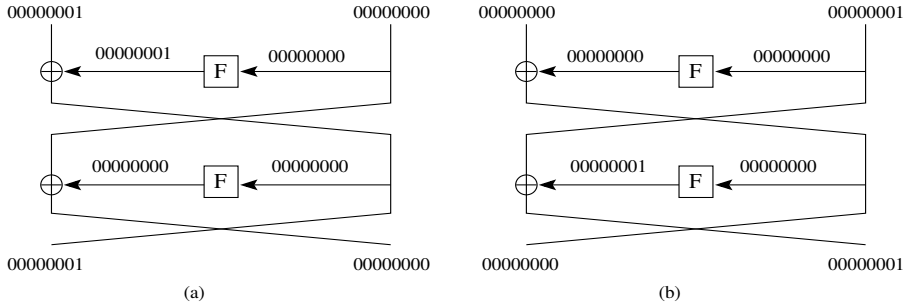


**Figure 2. Dual two-round iterative linear relations for CAST-128.**

The bias of the linear approximation $0 \rightarrow 1$ for each S-box was computed separately, and is $2^{-5}$ for all four S-boxes.

Notice that since we approximate only output bits from the S-boxes, our linear relations are not affected by the subkeys $K_{mi}$ and $K_{ri}$ at the input to the round functions.

We have focused efforts on searching for iterative linear relations, which means linear distinguishers that can be concatenated with themselves. We arrive at the 2-round iterative linear relations in Fig. 2(a) and Fig. 2(b), with one active $F$ function and four active S-boxes for every two rounds. Notice that each linear distinguisher in (2) holds for any of the round function types $f_1$, $f_2$ and $f_3$ (independent of the combination of $+$, $-$ and $\oplus$), since Fig. 2 exploits only the least significant bit in the linear approximations. Thus, changing the order of round functions or permuting the order of these arithmetical operations do not matter.

These linear distinguishers applied to $2t$-round CAST-128 allow us to distinguish these reduced-round ciphers from a random permutation (with the same block size). For instance, Fig. 2(a) applied to $2t$ rounds leads to the linear relation $(L_0 \oplus L_{2t}) \cdot \mathtt{00000001_x} = 0$ and Fig. 2(b) leads to $(R_0 \oplus R_{2t}) \cdot \mathtt{00000001_x} = 0$.

Since the block size of CAST-128 is only $64$ bits, and the computed bias (using the Piling-Up lemma) of Fig. 2(a) is $2^3 \cdot (2^{-5})^4 = 2^{-17}$, a distinguish-from-random attack on two rounds requires $2^{37}$ KP (using Matsui's estimate $8 * (\text{bias})^{-2}$ in [20]). For three rounds, one could use Fig. 2(b), at the same cost as for two rounds.

A key-recovery attack using one of the 2-round distinguishers in Fig. 2 requires guessing 37 subkey bits at once: five bits for the rotation subkey ($K_{mi}$) and 32 bits for the addition subkey ($K_{ri}$). For example, an attack on 3-round CAST-128, using two rounds of

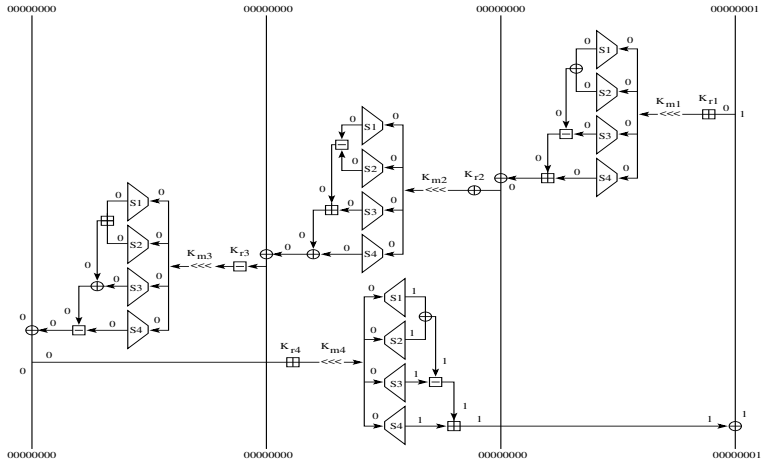


**Figure 3. One quad-round iterative linear relation for CAST-256.**

Fig. 2(a) would recover $K_{m3}$ and $K_{r3}$ at the cost of $2^{37} \cdot 2^{37} = 2^{74}$ one-round decryptions, or $2^{74}/3 \approx 2^{72.5}$ 3-round computations.

## 5. Linear Analysis of CAST-256

The same S-boxes of CAST-128 are also used in CAST-256. Likewise, the same three kinds of round functions of CAST-128 are applied in groups of four rounds called quad-rounds. Thus, we use in CAST-256 the same bit masks used in our analysis of CAST-128. It is interesting to observe that [5] already predicted linear approximations with nonzero output masks only. But, the authors of [5] did not discuss this issue further.

The first linear relations we have for CAST-256 are 4-round iterative (Fig.3) which stands for one quad-round. Only one (out of four) round is active, and all four $f_i$ functions of this round are active. Notice that relations in Fig. 3 can be applied to the inverse quad-rounds as well (and with the same bias).

We computed the bias for the linear approximation with masks $(\texttt{00}_{\text{x}}, \texttt{00000001}_{\text{x}})$, for each of the four S-boxes $S_1$ until $S_4$. The bias is exactly $2^{-5}$ for all of them.

We could have constructed alternative iterative linear relations, with higher-order bits set such as $\texttt{00000002}_{\text{x}}$ or $\texttt{00000003}_{\text{x}}$. For the bit mask $\texttt{00000002}_{\text{x}}$ all S-boxes present bias $2^{-5}$, and for $\texttt{00000003}_{\text{x}}$ the bias are $2^{-6}$, $2^{-5}$, $2^{-6}$ and $2^{-7}$ for S-boxes $S_1$ to $S_4$, respectively. These masks lead to a decrease of $2^{-3}$ in the combined bias due carry and borrow bits in the modular addition and subtraction operations in the round functions. For instance, for one full round, using mask $\texttt{00000002}_{\text{x}}$, the bias becomes $2^4 \cdot 2^{-5} \cdot 2^{-5} \cdot 2^{-5} \cdot 2^{-5} \cdot 2^{-3} = 2^{-19}$. The factor $2^{-3}$ is due to the approximation of modular subtraction and addition with mask $\texttt{00000002}_{\text{x}}$. Thus, we got no significant advantage in using these alternative bit masks instead of $\texttt{00000001}_{\text{x}}$.

The linear relation in Fig. 3 can be used to distinguish a number of quad-rounds of CAST-256 from a random permutation (with the same block size). Similar to Sect. 4., one can derive a linear relation such as $(D \oplus H) \cdot \texttt{00000001}_{\text{x}} = 0$, where $(A, B, C, D)$ denotes a plaintext block, and $(E, F, G, H)$ denotes a ciphertext block after a number of quad-rounds.

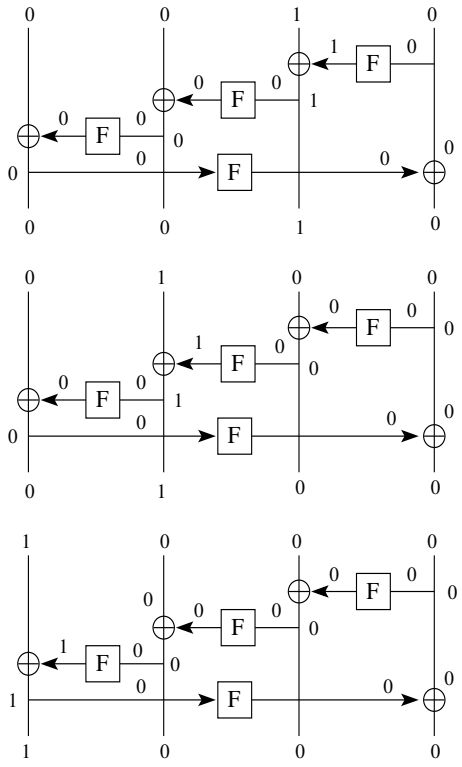


**Figure 4. One quad-round iterative linear relations for CAST-256.**

### 5.1. Key-Recovery Attacks

Similar to Sect. 4., a key recovery attack using Fig. 3 or similar distinguishers in Fig.4 can recover $37$ subkey bits from a round (function) at the end of the distinguisher. For instance, an attack on $5$-round CAST-256 using a $4$-round distinguisher would recover $K_{m5}$ and $K_{r5}$, using $2^{37}$ KP, and would take $2^{37} \cdot 2^{37}$ one-round decryptions, or $2^{74}/5 = 2^{71.7}$ 5-round computations.

Similarly, attacking 9-round CAST-256 using an 8-round distinguisher would require $2^{69}$ KP, and $2^{69} \cdot 2^{37}$ one-round decryptions or $2^{106}/9 \approx 2^{103}$ 9-round computations.

## 6. Differential Cryptanalysis: negative results

The four S-boxes $S_1$, $S_2$, $S_3$ and $S_4$ are injective, but non-surjective mappings. We have further computed the values of the round functions $f_1$, $f_2$ and $f_3$ for all 32-bit input values without the $K_{mi}$ and $K_{ri}$ subkeys (because these key-dependent operations only permute the input values).

More precisely, only $2714938431 \approx 2^{31.34}$ values, out of the $2^{32}$, show up at the output to $f_1$, namely, $1580028865 \approx 2^{30.56}$ are missing. Analogously, $1580108708 \approx 2^{30.56}$, out of the $2^{32}$ output values from $f_2$, are missing. Finally, $1580054165 \approx 2^{30.56}$, out of the $2^{32}$ output values from $f_3$, are missing i.e. only $2714913131 \approx 2^{31.34}$ values can appear.

Thus, whatever the value of $K_{mj}$ and $K_{rj}$, none of $f_i$, $1 \leq i \leq 3$, is surjective. On one hand, this property protects CAST-128 and CAST-256 from conventional multiset/square attacks [11] and impossible differential (ID) attacks [16] since a nonzero input

xor-difference can cause a zero output difference (although with a small probability). This last fact may explain the absence of ID distinguishers claimed by Biham in [6]. This same fact, namely the existence of some xor-difference $\Delta \neq 0$ that can cause a zero output xor-difference with nonzero probability for all three round functions, means the existence of iterative differential characteristics which have a dual form to that of linear relation (2), namely, with nonzero input xor-difference but zero output xor-difference.

Nonetheless, unlike the linear relation (2), the differential characteristic is very dependent on $K_{mj}$ and $K_{rj}$, since they control the actual input differences to $f_i$. In order to bypass these subkeys and minimize the number of active S-boxes, suppose we could find an input difference of the form $\Delta = 2^t$, for some $0 \leq t \leq 31$ (the lowest possible Hamming weight). In particular, for $i = 31$ the characteristic would have the form $\texttt{80000000}_{\texttt{x}} \xrightarrow{f_i} \texttt{00000000}_{\texttt{x}}$ for the round function ($f_i$). The difference $\texttt{80000000}_{\texttt{x}}$ was chosen to bypass the modular addition with a subkey $K_{mj}$ prior to the S-boxes. The reason for a 1-bit difference is the key-dependent rotation with a subkey $K_{rj}$. Using a 1-bit difference means that whatever the rotation amount $K_{rj}$ only one S-box would be active, because only one S-box would have a nonzero input difference. But, since the S-boxes are injective, it is not possible to affect only one S-box and expect a nonzero output difference. At least two S-boxes have to be active per $f_i$ function.

We have further analyzed differences of the form $2^i \oplus 2^j$, for $i \neq j$, $0 \leq i, j \leq 31$, but we could only find a few pairs of texts with an xor-difference that affects only two S-boxes (independent of $K_{mj}$ and $K_{rj}$), and generates a zero output difference for all three $f_i$. These differences have a probability between $77 \cdot 2^{-31} \approx 2^{-24.73}$ and $2 \cdot 2^{-31} = 2^{-30}$.

Nonetheless, it would still require particular values for $K_{mj}$ and $K_{rj}$ in order to preserve the difference. Further, the successive round functions alternate between $f_1$, $f_2$ and $f3$, and the nonzero input differences with high probability are distinct.

Thus, our preliminary differential analysis indicate that iterative differential distinguishers, using input differences with Hamming weight one, two, three or four do not lead to a better attack than linear cryptanalysis.

## 7. Conclusion

This paper described linear distinguishers and attacks for CAST-128 and CAST-256. As far as we are aware of, this paper reports the first known-plaintext attack on reduced-round versions of these ciphers.

Table 1 summarizes the known-plaintext attacks on CAST-128 and CAST-256. For CAST-128 we assume the key is larger than 73 bits. For CAST-256 we assume the key sizes specified for the AES Process (128, 192 or 256 bits).

Our linear analyses exploited only the output bits of each $8 \times 32$-bit S-box, and consequently, only the output of the round functions. Our linear masks exploit only approximations involving the least significant bit of the output of S-boxes and round functions. Therefore, these approximations hold for all of the different round functions, $f_i$, $1 \leq i \leq 3$, effectively bypassing the mixing of modular addition, exclusive-or and modular subtraction.

Since our linear masks exploit the least significant bit of some data value, we expect no significant linear hull effect [25].

**Table 1. Summary of linear attacks on CAST-128 and CAST-256.**

| Cipher | # Rounds | Data (KP)/ Memory | Time | Comments |
|---|---|---|---|---|
| CAST-128 | 2 | $2^{37}$ | $2^{37}$ | Fig.(2)(a); distinguishing attack |
| | 3 | $2^{37}$ | $2^{37}$ | Fig.(2)(b); distinguishing attack |
| | 3 | $2^{37}$ | $2^{72.5}$ | Fig.(2)(a); 37 key bits recovered |
| CAST-256 | 4 | $2^{37}$ | $2^{37}$ | distinguishing attack |
| | 5 | $2^{37}$ | $2^{71.7}$ | 37 key bits recovered |
| | 8 | $2^{69}$ | $2^{69}$ | distinguishing attack |
| | 9 | $2^{69}$ | $2^{103}$ | 37 key bits recovered |
| | 12 | $2^{101}$ | $2^{101}$ | distinguishing attack |

We leave as open problems the potential combination of linear and differential characteristics, and other attack techniques to improve the results in this paper.

We have analysed the Algebraic Normal Form (ANF) of each output bit of each $8 \times 32$-bit S-box of CAST-128 and CAST-256, and we have verified that each such Boolean function has nonlinear order exactly $4$ (see Appendix 8.).

Further, according to Courtois-Pieprzyk [10] and Biryukov-DeCanniere [7], one might expect $\binom{40}{0} + \binom{40}{1} + \binom{40}{2} - 2^8 = 780 + 40 + 1 - 256 = 565$ quadratic equations for $8 \times 32$-bit S-boxes. We have computed all of these quadratic equations, and confirmed these predictions (see example in Appendix8.). It is left as an open problem how to exploit these quadratic equations in an effective (algebraic) attack on CAST-128 and CAST-256.


## Acknowledgement

We would like to thank Vincent Rijmen for useful suggestions that improved the presentation of this paper.


## References


[1] AES, *The Advanced Encryption Standard Development Process*, 1997, http://csrc.nist.gov/encryption/aes/

[2] C.M. Adams, *The CAST-128 Encryption Algorithm*, RFC 2144, May 1997.

[3] C.M. Adams, *The CAST-256 Encryption Algorithm*, 1st AES Conference, California, USA, Aug. 1998, http://csrc.nist.gov/encryption/aes/

[4] C.M. Adams, *Constructing Symmetric Ciphers using the CAST Design Procedure*, Designs, Codes, and Cryptography, 12:(3), Nov. 1997, 283–316.

[5] C.M. Adams, H.M. Heys, S.E. Tavares, M. Wiener, *An Analysis of the CAST-256 Cipher*, http://www.securitytechnet.com/resource/crypto/algorithm/block/cast256.ps

[6] E. Biham, *A Note on Comparing the AES Candidates*, The AES Development Process, http://csrc.nist.gov/encryption/aes/round1/conf2/papers/biham2.pdf

[7] A. Biryukov, C. De Cannière, *Block Ciphers and Systems of Quadratic Equations*, 10th Fast Software Encryption Workshop, T. Johansson, Ed., Springer-Verlag, LNCS 2887, 2003, 274–289.

[8] U. Blöcher, M. Dichtl, *Problems with the Linear Cryptanalysis of DES using More than One Active S-box per Round*, 1st Fast Software Encryption Workshop, R. Anderson,Ed., Springer-Verlag, LNCS 809, 1994, 256–274.

[9] L. Brown, J. Pieprzyk, *Introducing the New LOKI97 Block Cipher*, 1st AES Conference, California, USA, Aug. 1998, http://csrc.nist.gov/encryption/aes/

[10] N.T. Courtois, J. Pieprzyk, *Cryptanalysis of Block Ciphers with Overdefined Systems of Quadratic Equations*, Adv. in Cryptology, Asiacrypt'02, Y. Zheng,Ed., Springer-Verlag, LNCS 2501, 2002, 267–287.

[11] J. Daemen, L.R. Knudsen, V. Rijmen, *The Block Cipher SQUARE*, 4th Fast Software Encryption Workshop, E. Biham,Ed., Springer-Verlag, LNCS 1267, 1997, 149–165.

[12] J. Daemen, V. Rijmen, *The Design of Rijndael – AES – The Advanced Encryption Standard*, Springer-Verlag, 2002.

[13] GnuPG, Gnu Privacy Guard, http://www.gnupg.org/(en)/features.html

[14] H.M. Heys, S.E. Tavares, *On the Security of the CAST Encryption Algorithm*, Canadian Conference on Electrical and Computer Engineering, 1994, 332–335.

[15] J. Kelsey, B. Schneier, D. Wagner, *Related-Key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X. NewDES, RC2 and TEA*, Information and Commmunications Security, ICICS'97, First International Conference Proceedings, Springer-Verlag, Nov. 1997, 233–246.

[16] L.R. Knudsen, *DEAL – a 128-bit Block Cipher*, Tech Report #151, Univ. of Bergen, Dept. of Informatics, Norway, Feb. 1998.

[17] L.R. Knudsen, *Weaknesses in LOKI97*, 1999, http://csrc.nist.gov/encryption/aes/

[18] L.R. Knudsen, J.E. Mathiassen, *A Chosen-Plaintext Linear Attack on DES*, 7th Fast Software Encryption Workshop, B. Schneier, Ed., Springer-Verlag, LNCS 1978, 2000, 262–272.

[19] L.R. Knudsen, V. Rijmen, *Ciphertext-Only Attack on Akelarre*, Cryptologia, vol. XXIV, no. 2, Apr. 2000, 135–147.

[20] M. Matsui, *Linear Cryptanalysis Method for DES Cipher*, Adv. in Cryptology, Eurocrypt'93, T. Helleseth,Ed., Springer-Verlag, LNCS 765, 1994, 386–397.

[21] M. Matsui, *On Correlation Between the Order of S-boxes and the Strength of DES*, Adv. in Cryptology, Eurocrypt'94, A. De Santis,Ed., Springer-Verlag, LNCS 950, 1995, 366–375.

[22] M. Matsui, A. Yamagishi, *A New Method for Known-Plaintext Attack of FEAL Cipher*, Adv. in Cryptology, Eurocrypt'92, R.A. Rueppel,Ed., Springer-Verlag, LNCS 658, 1993, 81–91.

[23] S. Moriai, T. Shimoyama, T. Kaneko, *Higher-Order Differential Attack of a CAST cipher*, Fast Software Encryption, 5th International Workshop Proceedings, Springer-Verlag, 1998, 17–31.

[24] NBS, *Data Encryption Standard (DES)*, FIPS PUB 46, Federal Information Processing Standards Publication 46, U.S. Department of Commerce, Jan. 1977.

[25] K. Nyberg, *Linear Approximation of Block Ciphers*, Adv. in Cryptology, Eurocrypt'94, A. De Santis,Ed., Springer-Verlag, LNCS 950, 1995, 439–444.

[26] V. Rijmen, B. Preneel, Erik De Win, *On Weaknesses of Non-Surjective Rounds Functions*, Designs, Codes and Cryptography, vol. 12, 1997, 253–266.

[27] R.L. Rivest, M.J.B. Robshaw, R. Sidney, Y.L. Yin, *The RC6 Block Cipher*, 1st AES Conference, California, USA, Aug. 1998, http://csrc.nist.gov/encryption/aes/

[28] B. Schneier, J. Kelsey, *Unbalanced Feistel Networks and Block Cipher Design*, 3rd Fast Software Encryption Workshop, D. Gollmann,Ed., Springer-Verlag, LNCS 1039, 1996, 121–144.

[29] A.A. Selçuk, *On Bias Estimation in Linear Cryptanalysis*, Progress in Cryptology - INDOCRYPT 2000, B. Roy, E. Okamoto, Eds., Springer-Verlag, LNCS 1977, 2000, 52–66.

[30] J. Sung, J. Kim, C. Lee, S. Hong, *Related-Cipher Attacks on Block Ciphers with Flexible Number of Rounds*, Western European Workshop on Research in Cryptlogy, WEWoRC 2005, C. Wolf, S. Lucks, P.-W. Yau, Eds., Lecture Notes in Informatics (LNI), P-74.

[31] Wikipedia, *http://en.wikipedia.org/wiki/GNU_Privacy_Guard*

## 8. appendix

Let $X = (x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0)$ denote the input, and $Y = (y_{31}, \ldots, y_0)$ denote the output of such S-boxes. This section provides the Algebraic Normal Form (ANF) of $y_{31}$ of S-box $S_1$ of CAST-128 and CAST-256 as an example of the output Boolean functions of the $8 \times 32$-bit S-boxes.

$$y_{31} = x_0+x_0x_1+x_0x_1x_2+x_3+x_0x_3+x_0x_1x_3+x_2x_3+x_1x_2x_3+x_1x_4+x_0x_1x_4+ x_0x_3x_4 + x_2x_5 + x_1x_2x_5 + x_1x_3x_5 + x_0x_4x_5 + x_6 + x_0x_1x_6 + x_2x_6 + x_1x_2x_6 + x_3x_6 + x_1x_3x_6+x_0x_1x_3x_6+x_2x_3x_6+x_4x_6+x_0x_4x_6+x_1x_4x_6+x_0x_1x_4x_6+x_2x_4x_6+x_0x_2x_4x_6+ x_0x_3x_4x_6+x_2x_3x_4x_6+x_2x_5x_6+x_0x_2x_5x_6+x_1x_2x_5x_6+x_3x_5x_6+x_2x_3x_5x_6+x_4x_5x_6+ x_1x_4x_5x_6 + x_0x_7 + x_1x_7 + x_0x_1x_7 + x_0x_1x_2x_7 + x_3x_7 + x_0x_3x_7 + x_1x_2x_3x_7 + x_4x_7 + x_0x_4x_7 + x_0x_2x_4x_7 + x_2x_3x_4x_7 + x_0x_5x_7 + x_1x_5x_7 + x_0x_1x_5x_7 + x_1x_2x_5x_7 + x_3x_5x_7 + x_4x_5x_7 + x_1x_4x_5x_7 + x_3x_4x_5x_7 + x_1x_6x_7 + x_0x_1x_6x_7 + x_0x_2x_6x_7 + x_3x_6x_7 + x_1x_3x_6x_7 + x_2x_3x_6x_7+x_4x_6x_7+x_0x_4x_6x_7+x_3x_4x_6x_7+x_0x_5x_6x_7+x_1x_5x_6x_7+x_3x_5x_6x_7+x_4x_5x_6x_7.$$

An example of quadratic equation for S-box $S_1$ is

$$1 + x_1 + x_2 + x_4 + y_0 + y_1 + y_2 + y_3 + y_5 + y_6 + y_7 + y_8 + y_{10} + y_{11} + y_{13} + y_{14} + y_{15} + y_{16} + y_{18} + y_{20} + y_{21} + y_{22} + y_{23} + y_{28} + y_{29} + y_{31} + x_0x_2 + x_0x_5 + x_0x_6 + x_0y_2 + x_0y_6 + x_0y_7 + x_0y_8 + x_0y_9 + x_0y_{10} + x_0y_{11} + x_0y_{12} + x_0y_{13} + x_0y_{15} + x_0y_{17} + x_0y_{18} + x_0y_{19}+x_0y_{21}+x_0y_{22}+x_0y_{25}+x_0y_{30}+x_0y_{31}+x_1y_0+x_1y_4+x_1y_5+x_1y_6+x_1y_7+x_1y_9+ x_1y_{11} + x_1y_{12} + x_1y_{16} + x_1y_{17} + x_1y_{20} + x_1y_{22} + x_1y_{24} + x_1y_{26} + x_1y_{27} + x_1y_{28} + x_1y_{30} + x_1y_{31}+x_2x_3+x_2y_0+x_2y_4+x_2y_6+x_2y_7+x_2y_8+x_2y_{11}+x_2y_{12}+x_2y_{14}+x_2y_{15}+x_2y_{18}+ x_2y_{20}+x_2y_{25}+x_2y_{26}+x_2y_{29}+x_2y_{31}+x_3x_4+x_3x_6+x_3y_1+x_3y_3+x_3y_4+x_3y_5+x_3y_6+ x_3y_7 + x_3y_9 + x_3y_{10} + x_3y_{11} + x_3y_{12} + x_3y_{14} + x_3y_{15} + x_3y_{21} + x_3y_{24} + x_3y_{25} + x_3y_{27} + x_3y_{30}+x_4x_7+x_4y_1+x_4y_2+x_4y_3+x_4y_6+x_4y_8+x_4y_9+x_4y_{11}+x_4y_{14}+x_4y_{18}+x_4y_{21}+ x_4y_{22}+x_4y_{23}+x_4y_{25}+x_4y_{27}+x_4y_{30}+x_5x_7+x_5y_1+x_5y_{13}+x_5y_{15}+x_5y_{21}+x_5y_{28} = 0.$$