

# Fault-based Attacks on the Bel-T Block Cipher Family

Philipp Jovanovic and Ilia Polian  
 University of Passau, 94032, Germany  
 {philipp.jovanovic|ilia.polian}@uni-passau.de

**Abstract**—We present the first fault-based attack on the Bel-T block cipher family which has been adopted recently as a national standard of the Republic of Belarus. Our attack successfully recovers the secret key of the 128-bit, 192-bit and 256-bit versions of Bel-T using 4, 7 and 10 fault injections, respectively. We also show the results from our comprehensive simulation-based experiments.

**Keywords**—fault-based cryptanalysis, fault injection, block cipher

## I. INTRODUCTION

In this paper, we investigate the vulnerability of the block cipher family Bel-T to *fault-based attacks* [1] which belong to the category of active side-channel cryptanalysis. Bel-T has been approved as a standard of Republic of Belarus in 2011. Its specification in Russian language is available from its developer’s web site, the Research Institute for Applied Problems of Mathematics and Informatics of the Belarussian State University [2]. We are not aware of an English-language specification of this cipher, nor of any published results on its security. Bel-T follows the Lai-Massey scheme [3], which has similarities with both substitution-permutation networks and Feistel networks. Earlier ciphers designed according to this scheme include IDEA [3] and its extension IDEA NXT, also known as FOX [4]. Fault-based attacks are known for both IDEA [5] and FOX [6], but they utilize specific features of these ciphers and not the Lai-Massey construction itself and are thus not applicable to Bel-T.

The remainder of the paper is organized as follows. Section II provides a specification of Bel-T which is the first English-language description of this cipher as far as we know. The new fault-based attack is described in Section III and the simulation results are presented in Section IV. Finally, Section V concludes the paper.

## II. SPECIFICATION

For reasons of consistency, we describe the specification of Bel-T using the same notation as in the original (Russian-language) document [2]. In particular, given  $u, v \in \{0, 1\}^n$ ,  $u \oplus v$  stands for the bit-wise addition modulo 2 (exclusive-or) of  $u$  and  $v$ , and  $u \boxplus v$  and  $u \boxminus v$ , respectively, stand for the arithmetical addition

and subtraction of  $u$  and  $v$  modulo  $2^n$ , where  $u$  and  $v$  are interpreted as unsigned integers.

Bel-T is a block cipher which encrypts a 128-bit plaintext  $X$  using a 256-bit value  $\theta = \theta_1 \parallel \dots \parallel \theta_8$ , with 32-bit words  $\theta_i$  for  $1 \leq i \leq 8$ , to obtain a 128-bit ciphertext  $Y$ . The Bel-T family consists of three ciphers which employ secret keys of different lengths (128 bits, 192 bits and 256 bits) and are identical otherwise. We call these versions Bel-T-128, Bel-T-192 and Bel-T-256, respectively:

- Bel-T-256: The value  $\theta$  is identical to the 256-bit secret key.
- Bel-T-192: The words  $\theta_1, \dots, \theta_6$  correspond to the 192-bit secret key, and the remaining two words are set to  $\theta_7 := \theta_1 \oplus \theta_2 \oplus \theta_3$  and  $\theta_8 := \theta_4 \oplus \theta_5 \oplus \theta_6$ .
- Bel-T-128: The words  $\theta_1, \dots, \theta_4$  correspond to the 128-bit secret key. The remaining four words are set to  $\theta_5 := \theta_1$ ,  $\theta_6 := \theta_2$ ,  $\theta_7 := \theta_3$  and  $\theta_8 := \theta_4$ .

Note that  $\theta$  is a 256-bit value in all three versions.

Encryption, see Figure 1, is denoted by  $Y = F_\theta(X)$ , and decryption is denoted by  $Y = F_\theta^{-1}(X)$ . Both are organized in eight rounds.

```

1  a := X1; b := X2; c := X3; d := X4;
2  for i = 1, ..., 8 do
    1) b := b ⊕ G5(a ⊞ K7i-6);
    2) c := c ⊕ G21(d ⊞ K7i-5);
    3) a := a ⊞ G13(b ⊞ K7i-4);
    4) e := G21(b ⊞ c ⊞ K7i-3) ⊕ ⟨i⟩32;
    5) b := b ⊞ e;
    6) c := c ⊞ e;
    7) d := d ⊞ G13(c ⊞ K7i-2);
    8) b := b ⊕ G21(a ⊞ K7i-1);
    9) c := c ⊕ G5(d ⊞ K7i);
    10) swap a and b;
    11) swap c and d;
    12) swap b and c;
3  Y := b || d || a || c;
4  return Y;
```

Fig. 1. Encryption of  $X = X_1 \parallel X_2 \parallel X_3 \parallel X_4$

The rounds use different sets of *round keys* and are identical otherwise. There are 56 round keys of 32 bits each and in round  $i \in \{1, \dots, 8\}$  seven round keys  $K_{7i-j}$ , with  $0 \leq j \leq 6$ , are used. Their order is shown in Table I, for encryption from top to bottom

and for decryption the other way round. Also note the different ordering of the  $K_{7i-j}$ , during encryption (top) and decryption (bottom).

TABLE I  
KEY USAGE IN BEL-T

	$i$	$K_{7i-6}$	$K_{7i-5}$	$K_{7i-4}$	$K_{7i-3}$	$K_{7i-2}$	$K_{7i-1}$	$K_{7i}$	
Encryption	1	$\theta_1$	$\theta_2$	$\theta_3$	$\theta_4$	$\theta_5$	$\theta_6$	$\theta_7$	Decryption
	2	$\theta_8$	$\theta_1$	$\theta_2$	$\theta_3$	$\theta_4$	$\theta_5$	$\theta_6$	
	3	$\theta_7$	$\theta_8$	$\theta_1$	$\theta_2$	$\theta_3$	$\theta_4$	$\theta_5$	
	4	$\theta_6$	$\theta_7$	$\theta_8$	$\theta_1$	$\theta_2$	$\theta_3$	$\theta_4$	
	5	$\theta_5$	$\theta_6$	$\theta_7$	$\theta_8$	$\theta_1$	$\theta_2$	$\theta_3$	
	6	$\theta_4$	$\theta_5$	$\theta_6$	$\theta_7$	$\theta_8$	$\theta_1$	$\theta_2$	
	7	$\theta_3$	$\theta_4$	$\theta_5$	$\theta_6$	$\theta_7$	$\theta_8$	$\theta_1$	
	8	$\theta_2$	$\theta_3$	$\theta_4$	$\theta_5$	$\theta_6$	$\theta_7$	$\theta_8$	
	$i$	$K_{7i}$	$K_{7i-1}$	$K_{7i-2}$	$K_{7i-3}$	$K_{7i-4}$	$K_{7i-5}$	$K_{7i-6}$	

Three mappings  $G_5$ ,  $G_{13}$  and  $G_{21} : \{0, 1\}^{32} \rightarrow \{0, 1\}^{32}$  are used, where  $G_r$  maps a 32-bit word  $u = u_1 \parallel u_2 \parallel u_3 \parallel u_4$ , with  $u_i \in \{0, 1\}^8$ , as follows:

$$G_r(u) = (H(u_1) \parallel H(u_2) \parallel H(u_3) \parallel H(u_4)) \lll r.$$

Here,  $H$  is the SBox specified in Table II and  $\lll r$  stands for a cyclical shift to the left by  $r$  positions. The value  $\langle i \rangle_{32}$  in line 4) of the encryption algorithm stands for the binary number of the round. The diagram in Figure 2 shows the functionality of a round. It also indicates how Bel-T can be implemented in hardware.

TABLE II  
THE BEL-T SBOX  $H$

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	B1	94	EA	C8	0A	08	F5	3B	36	6D	00	8E	58	4A	5D	E4
1	85	04	FA	9D	1B	B6	C7	AC	25	2E	72	C2	02	FD	CE	0D
2	5B	E3	D6	12	17	B9	61	81	FE	67	86	AD	71	6B	89	0B
3	5C	B0	C0	FF	33	C3	56	B8	35	C4	05	AE	D8	E0	7F	99
4	E1	2B	DC	1A	E2	82	57	EC	70	3F	CC	F0	95	EE	8D	F1
5	C1	AB	76	38	9F	E6	78	CA	F7	C6	F8	60	D5	BB	9C	4F
6	F3	3C	65	7B	63	7C	30	6A	DD	4E	A7	79	9E	B2	3D	31
7	3E	98	B5	6E	27	D3	BC	CF	59	1E	18	1F	4C	5A	B7	93
8	E9	DE	E7	2C	8F	0C	0F	A6	2D	DB	49	F4	6F	73	96	47
9	06	07	53	16	ED	24	7A	37	39	CB	A3	83	03	A9	8B	F6
A	92	BD	9B	1C	E5	D1	41	01	54	45	FB	C9	5E	4D	0E	F2
B	68	20	80	AA	22	7D	64	2F	26	87	F9	34	90	40	55	11
C	BE	32	97	13	43	FC	9A	48	A0	2A	88	5F	19	4B	09	A1
D	7E	CD	A4	D0	15	44	AF	8C	A5	84	50	BF	66	D2	E8	8A
E	A2	D7	46	52	42	A8	DF	B3	69	74	C5	51	EB	23	29	21
F	D4	EF	D9	E4	3A	62	28	75	91	14	10	EA	77	6C	DA	1D

Decryption is identical to encryption, see Figure 1, with one exception: round key  $K_{7i}$  is used in line 1) instead of  $K_{7i-6}$ ; round key  $K_{7i-1}$  is used in line 2) instead of  $K_{7i-5}$ ; and so forth. This reduces the complexity of the algorithm, in particular for a hardware implementation, where the same circuitry can be used for both encryption and decryption. This feature is also instrumental for the fault-based attack presented below.

### III. FAULT-BASED ATTACK ON BEL-T

The fault-based attack scenario assumes that the attacker has physical access to the device that performs the encryption, is capable to encrypt plaintexts of her choice

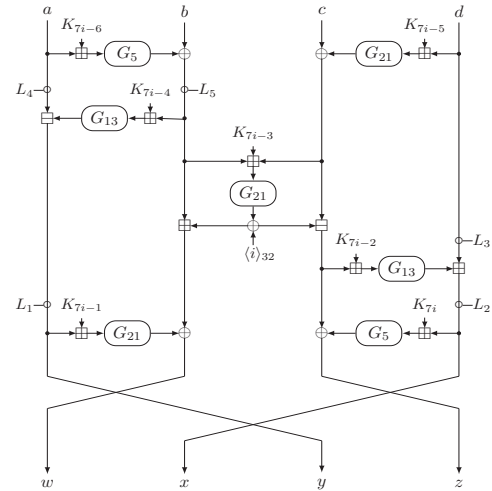


Fig. 2. The  $i$ -th Bel-T round [2] with fault locations  $L_j$

and observe the resulting ciphertext, to decrypt ciphertexts of her choice and observe the resulting plaintext, and to inject faults during encryption or decryption. The aim of the attacker is to recover the secret key  $\theta$ , which is stored (not directly accessible) within the device.

Let  $X$  be an arbitrary 128-bit plaintext and let  $Y$  be the 128-bit ciphertext calculated by the device in absence of any fault injections:  $Y = F_\theta(X)$ . The fault-based attack is conducted by series of fault injections, where “fault injection” refers to performing the encryption of the same plaintext  $X$  while injecting a transient fault and recording the faulty ciphertext  $Y^f$ . It is assumed that the secret key does not change during the whole attack, that is, all fault-affected encryptions are performed using the same  $\theta$  (and the same  $X$ ) as the fault-free encryption. Each fault injection determines several bits of  $\theta$ . Moreover, the same principle is applied to decryptions: Let  $\tilde{X}$  be an arbitrary ciphertext and let  $\tilde{Y} = F_\theta^{-1}(\tilde{X})$  be the plaintext obtained by decryption in absence of faults. Injecting faults during decryption results in deviating plaintexts  $\tilde{Y} = F_\theta^{-1,f}(\tilde{X})$ .

All faults used in our attack are applied during the last of the eight rounds of Bel-T. Before we describe the concrete attacks on Bel-T, we introduce the two fault models that are used in our analysis:

- The random fault model (RFM) assumes that an attacker can inject faults into an element of the state at a freely chosen position, such that its value switches to a random, unknown value.
- The chosen fault model (CFM) assumes that an attacker can inject faults into an element of the state at a freely chosen position, such that the value of the state element switches to a known value.

To perform our attack on Bel-T-128, we require only

4 RFM faults. For Bel-T-192 and Bel-T-256 we likewise need 4 RFM faults, but 3 respectively 6 additional CFM faults. We first describe the attack on Bel-T-128, because it also forms the basis for the fault attacks on Bel-T-192 and Bel-T-256. In the following, we denote the fault-free outputs at the end of encryption and decryption, respectively, by  $w, x, y$  and  $z$ , as in Figure 2, and the fault-affected outputs by  $w', x', y'$  and  $z'$ .

#### A. Fault Attack on Bel-T-128

We start our attack on Bel-T-128 with the aim to reconstruct subkey  $K_{7i-1}$  in the last round of the encryption ( $i = 8$ ), which corresponds to  $\theta_7 = \theta_3$ . For this purpose, we inject the first fault  $f_1$  (RFM) into the state at position marked  $L_1$  in Figure 2. Note that  $f_1$  can flip an arbitrary number of bits within the 32-bit word and the best results are achieved during filtering if  $f_1$  affects all of the 4 bytes in  $L_1$ . The value of  $f_1$  is not immediately known to the attacker. However, it can be derived by observing the correct and faulty ciphertexts  $Y$  and  $Y^{f_1}$  at positions  $y$  and  $y'$  of the cipher's output and calculating the XOR of these values:  $f_1 = y \oplus y'$ . The fault propagates through the key addition of  $\theta_7$ , the application of  $G_{21}$  and the XOR with the  $b$ -part of the state, and creates an XOR-difference  $w \oplus w'$ , which is also directly observable at the cipher's outputs. Since the value at  $L_1$  corresponds to  $y$  during a fault-free encryption, subkey  $\theta_7$  must obey the following formula:

$$G_{21}(L_1 \boxplus \theta_7) \oplus G_{21}((L_1 \oplus f_1) \boxplus \theta_7) = w \oplus w' \quad (1)$$

The XOR with the  $b$ -part of the state is ignored since it does not influence  $w \oplus w'$ . All values in the above equation are known except for  $\theta_7$ . Eq. 1 is checked for all  $2^{32}$  bit combinations of  $\theta_7$ , and all candidates satisfying the equation are collected in a set called  $\Theta_7$ .

Referring again to Figure 2, we observe that keys added at  $K_{7i} (= \theta_8 = \theta_4)$ , can be reconstructed analogously. Fault  $f_2$  is injected at position  $L_2$  during encryption in round 8 and we use the information on  $f_2 = x \oplus x'$  and  $z \oplus z'$  from the corresponding outputs for our analysis. Thereby, we can reduce the number of candidates for subkey  $\theta_8$  and store them in set  $\Theta_8$ .

For retrieval of the two missing subkeys  $\theta_1 (= \theta_5)$  and  $\theta_2 (= \theta_6)$ , we exploit the property that encryption and decryption of Bel-T are basically the same. We switch from encryption to decryption and attack again the last round ( $i = 1$ ). Looking at Table I, we see that at positions  $K_{7i}$  and  $K_{7i-1}$  subkeys  $\theta_1$  and  $\theta_2$  are added to the state. Thus, injecting faults  $f_3$  and  $f_4$  at positions  $L_1$  and  $L_2$  in decryption allows us to reconstruct the two

missing subkeys using the same approaches as above. The candidates are stored in sets  $\Theta_1$  and  $\Theta_2$ , respectively.

After that the candidates for the secret key  $\theta = \theta_1 \parallel \dots \parallel \theta_8 = \theta_1 \parallel \theta_2 \parallel \theta_7 \parallel \theta_8 \parallel \theta_1 \parallel \theta_2 \parallel \theta_7 \parallel \theta_8$  must be contained in  $\Theta_1 \times \Theta_2 \times \Theta_7 \times \Theta_8 \times \Theta_1 \times \Theta_2 \times \Theta_7 \times \Theta_8$ . In our experiments, this set was always sufficiently small to perform a brute-force search for the correct key. If not, we can reduce this set by repeating one or all of the above procedures and computing the intersection of the corresponding subkey candidate sets.

#### B. Fault Attack on Bel-T-192

The attack on Bel-T-192 starts with the same 4 fault injections and the subsequent analysis as in the Bel-T-128-case. Thus, we retrieve information on  $\theta_1, \theta_2, \theta_7 = \theta_1 \oplus \theta_2 \oplus \theta_3$  and  $\theta_8 = \theta_4 \oplus \theta_5 \oplus \theta_6$  and store them in sets  $\Theta_1, \Theta_2, \Theta_7$  and  $\Theta_8$ . Since  $\theta_3 = \theta_7 \oplus \theta_1 \oplus \theta_2$ , the set of subkey candidates  $\Theta_3$  is obtained from  $\Theta_1, \Theta_2$  and  $\Theta_7$  by collecting all XOR combinations of values from these three sets. The missing subkeys are two out of three from  $\theta_4, \theta_5$ , and  $\theta_6$ , since we already know the XOR of the latter three from  $\theta_8$ .

Our next target is the key added at position  $K_{7i-2}$  in the last round of encryption, which corresponds to  $\theta_6$ . The determination of this subkey cannot be achieved by injecting an RFM fault before the key addition of  $K_{7i-2}$ , because the difference propagates through the latter and  $G_{13}$  but the result is masked by the  $\boxplus$ -operation with the  $d$ -state, which is unknown at this point and therefore leads to an unpredictable outcome. We switch the fault model from RFM to CFM and assume that an attacker can inject faults which set a part of the state to a fixed and known value. We assume for simplicity that this value is 0, but the analysis works for any other value, as well. The CFM fault  $f_5$  is injected at position  $L_3$ , resetting the  $d$ -state at this point to 0. This allows us to build an equation for filtering  $\theta_6$ -candidates of the form

$$G_{13}(s \boxplus \theta_6) \boxplus 0 = x' \quad (2)$$

where  $s = G_5(x \boxplus \theta_8) \oplus z$ . Recall that we already reduced the number of  $\theta_8$ -candidates at this point. A search over all combinations of  $\theta_6$  and  $\theta_8$  candidates is therefore feasible, as long as the number of  $\theta_8$  values is not too large. During our experiments this was never the case though, as described later. Finally, we again store all  $\theta_6$  candidates which satisfy Eq. 2 in set  $\Theta_6$ .

Repeating the above approach for decryption is not necessary, since subkey  $\theta_3$  is added at position  $K_{7i-2}$ , see Table I, and we already restricted the number of candidates for the latter in our previous analysis.

Our next target is the key addition  $K_{7i-4}$ , where subkey  $\theta_4$  is processed during the last round of encryption. We need dual faults  $f_6$  and  $f_7$  at  $L_4$  and  $L_5$  which reset the particular state words to 0 and circumvent masking with unknown state elements  $a$  and  $b$ . The  $\theta_4$  values that satisfy the resulting filtering equation

$$0 \boxplus G_{13}(0 \boxplus \theta_4) = y' \quad (3)$$

are stored in set  $\Theta_4$ .

The last missing subkey  $\theta_5$  can be reconstructed from the knowledge of the candidates for  $\theta_4$ ,  $\theta_6$  and  $\theta_8 = \theta_4 \oplus \theta_5 \oplus \theta_6$  and all  $\theta_5$  candidates are stored in set  $\Theta_5$ . The complete key is again found by the brute force search in  $\Theta_1 \times \dots \times \Theta_8$ . In summary, a total of 7 faults is sufficient to reconstruct the secret key  $\theta$  for Bel-T-192.

### C. Fault Attack on Bel-T-256

Mounting the Bel-T-128 attack on Bel-T-256 (by injecting RFM faults  $f_1$ ,  $f_2$ ,  $f_3$ , and  $f_4$ ) we can reconstruct candidates for subkeys  $\theta_1$ ,  $\theta_2$ ,  $\theta_7$ , and  $\theta_8$ . Unlike for Bel-T-192, the remaining values  $\theta_3$ ,  $\theta_4$ ,  $\theta_5$  and  $\theta_6$  are independent from the already reconstructed ones. All following faults are CFMs in the last round of encryption or decryption, which set the attacked element(s) to 0. Using an approach similar to Bel-T-192 we can collect information on subkey  $\theta_6$ . Repeating the same attack on the last round of decryption, fault  $f_6$  at position  $L_3$  gives us candidates for  $\theta_3$ . The filtering equations for injections of faults  $f_5$  and  $f_6$  are of a similar shape as Eq. 2.

The last two subkeys  $\theta_4$  and  $\theta_5$  are reconstructed by dual fault injections  $f_7$  and  $f_8$ , and  $f_9$  and  $f_{10}$  at locations  $L_4$  and  $L_5$  during last round of encryption and decryption, respectively, see again Table I and Figure 2, with filtering equations similar to Eq. 3. In summary, 10 fault injections are sufficient to break Bel-T-256.

## IV. RESULTS

We performed 5,000 runs for each of the above attacks and recorded the sizes of the respective (sub)key candidate sets. Table III gives an overview on the results of our simulations. We listed only the results for the actual (sub)keys and omitted the information on those that are generated during the Bel-T “key-schedule”, like  $\theta_7$  and  $\theta_8$  in case of Bel-T-192, since they do not provide any additional insights. The results clearly show that the fault injections, as detailed in Section III, are sufficient to reduce the key space of each Bel-T variant such that a subsequent brute-force on the remaining  $\theta$ -candidates becomes feasible. The implementation of our attack was written in C/C++, but did not exploit parallelisation. All

experiments were performed on a workstation with an AMD Opteron 6172 Processor operating at 2.1 GHz. For the analysis of one instance (i.e. reconstruction of one key) without the subsequent brute-force, we measured average running times of 148.0 (Bel-T-128), 287.0 (Bel-T-192), and 687.0 (Bel-T-256) seconds, respectively.

TABLE III  
STATISTICS ON THE BINARY LOGARITHMS FOR THE NUMBER OF KEY CANDIDATES

		$\Theta$	$\Theta_1$	$\Theta_2$	$\Theta_3$	$\Theta_4$	$\Theta_5$	$\Theta_6$	$\Theta_7$	$\Theta_8$
Bel-T-128	min	0.00	0.00	0.00	0.00	0.00	-	-	-	-
	max	22.00	10.00	10.58	17.00	10.58	-	-	-	-
	avg	5.11	3.32	3.17	5.64	3.00	-	-	-	-
	med	4.58	1.00	1.00	1.00	1.00	-	-	-	-
Bel-T-192	min	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-	-
	max	40.00	10.32	10.00	17.58	0.00	19.17	9.58	-	-
	avg	10.06	3.32	3.00	7.71	0.00	11.26	2.81	-	-
	med	9.17	1.00	1.00	3.58	0.00	2.00	1.00	-	-
Bel-T-256	min	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	max	39.00	10.00	10.00	10.00	0.00	0.00	10.58	16.00	10.58
	avg	7.63	3.17	3.17	3.17	0.00	0.00	3.32	4.46	3.32
	med	7.00	1.00	1.00	1.00	0.00	0.00	1.00	1.00	1.00

## V. CONCLUSIONS

We presented a detailed fault analysis of the Bel-T block cipher family by showing for each of its three variants, differing by key sizes of 128-, 192- and 256-bit, how to successfully mount attacks requiring only 4, 7 and 10 faults, respectively. The attack against Bel-T-128 can be completely performed under a very weak fault model (RFM). The attacks on Bel-T-192 and Bel-T-256 using RFM faults recover only 96 and 128 bits of the 192- and 256-bit master key. Under a slightly stronger fault model (CFM) the security of the latter two variants collapses nevertheless and the entire secret key can be reconstructed. Our experiments show that all attacks yield compact sets of key candidates for which brute-force search is practical.

## REFERENCES

- [1] D. Boneh, R. DeMillo, and R. Lipton, “On the Importance of Elimination Errors in Cryptographic Computations,” *Journal of Cryptology*, vol. 14, pp. 101–119, 2001.
- [2] “Data Encryption and Integrity Algorithms,” Preliminary State Standard of Republic of Belarus (STB P 34.101.31–2007), <http://apmi.bsu.by/assets/files/std/belt-spec27.pdf>.
- [3] X. Lai and J. Massey, “A Proposal for a New Block Encryption Standard,” in *EUROCRYPT 90*, ser. LNCS, I. B. Damgrd, Ed., vol. 473. Springer, 1991, pp. 389–404.
- [4] P. Junod and S. Vaudenay, “FOX: A New Family of Block Ciphers,” in *Selected Areas in Cryptography*, ser. LNCS, H. Handschuh and M. Hasan, Eds., vol. 3357, 2005, pp. 114–129.
- [5] C. Clavier, B. Gierlichs, and I. Verbauwhede, “Fault Analysis Study of IDEA,” in *CT-RSA 2008*, ser. LNCS, T. Malkin, Ed., vol. 4964, 2008, pp. 274–287.
- [6] L. Breveglieri, I. Koren, and P. Maistri, “A Fault Attack Against the FOX Cipher Family,” in *Fault Diagnosis and Tolerance in Cryptography*, ser. LNCS, L. Breveglieri, I. Koren, D. Naccache, and J.-P. Seifert, Eds., vol. 4236, 2006, pp. 98–105.