256 bit Standardized Crypto for 650 GE – GOST Revisited*

Axel Poschmann, San Ling and Huaxiong Wang

Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
{aposchmann, lingsan, hxwang}@ntu.edu.sg

Abstract. The former Soviet encryption algorithm GOST 28147-89 has been standardized by the Russian standardization agency in 1989 and extensive security analysis has been done since. So far no weaknesses have been found and GOST is currently under discussion for ISO standardization. Contrary to the cryptographic properties, there has not been much interest in the implementation properties of GOST, though its Feistel structure and the operations of its round function are well-suited for hardware implementations. Our post-synthesis figures for an ASIC implementation of GOST with a key-length of 256 bits require only 800 GE, which makes this implementation well suitable for low-cost passive RFID-tags. As a further optimization, using one carefully selected S-box instead of 8 different ones -which is still fully compliant with the standard specifications!- the area requirement can be reduced to 651 GE.

Keywords: lightweight cryptography, ASIC, GOST

1 Introduction

Increasingly, everyday items are enhanced to pervasive devices by embedding computing power and their interconnection leads to Mark Weiser's famous vision of *ubiquitous computing* (ubicomp) [27], which is widely believed to be the next paradigm in information technology. Pervasiveness requires mass deployment which in turn implies harsh cost constraints on the used technology. The cost constraints imply in particular for Application Specific Integrated Circuits (ASICs) that power, energy, and area requirements must be kept to a minimum. One counter-argument might be that *Moore's Law* will provide abundant computing power in the near future. However, Moore's Law needs to be interpreted contrary here: rather than doubling of performance, the price for constant computing power halves each 18 months. This interpretation leads to interesting conclusions, because many foreseen applications require a minimum amount of computing power, but at the same time have extremely tight cost constraints

* The research was supported in part by the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03.

(e.g. RFID on consumer items). As a consequence these applications are not realized yet, simply because they do not pay off. Moore's law however halves the price for a constant amount of computing power every 18 months, and consequently enables such applications after a certain period of time. Therefore, a constant or even increasing demand for the cheapest (read lightweight) solutions can be foreseen.

There are physical limits for the chip area: the smaller the area of a chip, the higher the relative costs for handling, packing and cutting of each of the chips, and thus there exists an optimal minimal chip area. Sometimes in the literature it is concluded that there is no need to develop lightweight cryptographic algorithms, because Moore's Law will provide an ever growing amount of computing power for this minimal area. However, we disagree with this viewpoint due to the following points. Firstly, there are plenty of engineering optimisation efforts ongoing to ever minimize the cutting losses and improve other manufacturing steps. Thus the optimal minimal chip area is constantly shrinking. Secondly, and more importantly, there are many envisioned functionalities for pervasive devices, security being only one amongst them. Thus decreasing the area demand for cryptographic primitives, on which security solutions are based, will increase the available space for other functionalities, that are maybe more valued by the users. In fact for RFID-tags used in supply chains there is a strong demand for storage in order to store status information directly on the tag. Thirdly, the smaller the overhead for cryptographic primitives is, the more likely security functionalities will be deployed.

1.1 Previous Work

The demand for small hardware area implementations of cryptographic algorithms has been widely recognised and so different approaches have been published. A lightweight AES core requiring 3400 GE and more than 1000 clock cycles has been first published in [7]. A different implementation of the AES requires only 3100 GE and is more than six times faster than the previous one [9]. In [14] a different approach is followed: DES and DESX have been slightly modified to DESL/DESXL and yield a more compact implementation without scrutinizing the security. There are also block cipher designs from scratch that aim at lightweight hardware implementations. Most notably are here PRESENT [2],[20], which requires only 1000 GE and is currently under ISO standardization and the recently proposed KTANTAN family [5], that can be scaled down to 462 GE, with hardwired key and block size of only 32 bits though.

In this paper we follow the first approach of optimizing the implementation of an existing standardized block ciphers, but we shift our focus far back in time, even before the demand of lightweight cryptography has been recognised. Back in 1989 the Soviet Union has standardized the block cipher GOST 28147-89 as a counterpart to DES (sometimes it is also called the "Russian DES"). For the sake of simplicity in the following we use the term GOST as a synonym for the encryption algorithm GOST 28147-89, though GOST is an abbreviation for *Gosudarstvennyi Standard*, the meaning of *Government Standard* in russian

language. Over the past 20 years quite some security analysis on GOST has been published, which we will briefly summarize here.

In [13] it has been shown that using a related key differential characteristic, GOST can be distinguished from a random permutation with probability $1 - 2^{-64}$. The authors propose further attacks on reduced-round versions of GOST and on full GOST, but can only obtain 12 bits of the master key, which leaves another 244 bits. Kara has performed a reflection attack on the full-round GOST in [12]. The attack assumes that the S-boxes are bijective and works only on a subset of approximately 2^{224} keys. Furthermore it requires 2^{32} chosen plaintexts and has a time complexity of 2^{192} steps, which is impractical.

A plain differential cryptanalysis can break up to 13 rounds of GOST using 2^{51} chosen plaintexts [23]. Combining this result with a related-key attack 21 rounds of GOST can be broken using 2^{56} chosen plaintexts [23].

The iterated structure of GOST can be exploited by slide attacks and in [1] a method is presented to break reduced-round versions of GOST with time complexity of 2^{63} encryptions. In case the S-boxes are known an adversary can break up to 30 rounds and 24 rounds if they are not known. In summary we can conclude that despite a considerable cryptanalytic effort has been spent over the past 20 years, GOST is still not broken and provides a security level of 256 bits.

Contrary to the cryptographic properties, there has not been much interest in the implementation properties of GOST. Software implementations of GOST have been presented in [18], but to the best of our knowledge there have been no hardware implementations of GOST reported so far. Therefore one contribution of this article is to provide lightweight hardware implementation details of GOST. Furthermore, we exploit the fact that the S-boxes in GOST can be chosen freely and propose to use the same S-box that has been used in PRESENT [2] to further decrease the area footprint. Our results reveal that GOST is well suited as an encryption algorithm for passive RFID-tags.

1.2 Outline

The remainder of this article is organized as follows: In the next Section we are going to briefly introduce the GOST encryption algorithm. Since the S-boxes are not specified in the original standard, we discuss the selection of an appropriate approach for the selection of S-boxes in the subsequent Section 3. There we also compare the linear and differential properties of the S-boxes as used by the Central Bank of Russian Federation and the PRESENT S-box. We propose a standard conform variant of GOST that uses the cryptographically strong PRESENT S-box. The hardware implementation results of both variants are presented in Section 4. Finally, this paper is concluded in Section 5.

2 Description of the GOST encryption algorithm

The former Soviet encryption standard GOST 28147-89 was published in [17]. There is an IETF draft [6] on GOST and GOST is currently discussed for inclusion into the ISO/IEC Standard 18033-3 on Block Ciphers [11]. GOST has a

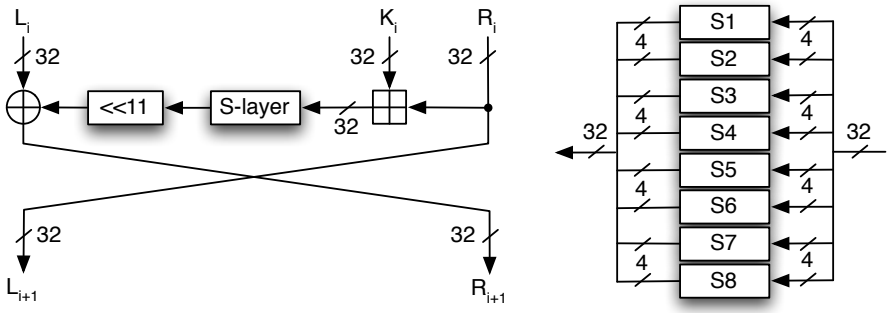

Fig. 1. One round of the Feistel network of GOST (left) and a detailed view of the S-layer (right).

block size of 64 bits and a key size of 256 bits. Its overall structure is a two branch Feistel network with 32 rounds and its inner roundfunction F consists of an integer addition, a non-linear substitution layer and an 11 bits left rotation. Let us denote the 64-bit data state of GOST by $STATE_i = L_i||R_i$, where $||$ denotes concatenation, then $STATE_0$ is the plaintext and $STATE_{32}$ the ciphertext. The integer addition adds the 32 bits roundkey K_i to the right halve of the state R_i, the result is then substituted by eight 4-bit to 4-bit S-box look-up tables and finally rotated by 11 bits to the left. The result of the roundfunction $F(K_i, R_i)$ is XORed to the left halve of the state L_i and is stored as the right halve of the subsequent round R_{i+1} , while R_i is stored without any modifications as L_{i+1}. In a formal notation we have

$$L_{i+1} = R_i,$$
$$R_{i+1} = L_i \oplus (S(K_i + R_i \bmod 2^{32}) \lll 11),$$

where $\oplus$ denotes a bitwise exclusive OR and $\lll a$ a rotation to the left by a bits. In the final round, the halves are not swapped, i.e. $R_{32} = R_{31}$ and $L_{32} = L_{31} \oplus (S(K_{31} + R_{31} \bmod 2^{32}) \lll 11)$. This enables to use the same hardware with the reverse round-key order for decryption. Figure 1 depicts one round of GOST graphically.

There is no real key schedule for GOST, instead the 256-bit key K is considered as eight 32-bit keys, $K = K_0||K_1||K_2||K_3||K_4||K_5||K_6||K_7$. For rounds $0 \leq r \leq 23$ the roundkey K_i is derived as $K_i = K_{(r \bmod 8)}$ and for the last eight rounds $24 \leq r \leq 31$ the order is reversed, i.e. $K_i = K_{7-(r \bmod 8)}$. Table 1 provides an overview of the roundkeys used in every round.

3 The choice of a set of S-boxes

The GOST standard does not specify a set of S-boxes. In fact, one aim of the designers was to have an encryption algorithm with a flexible security level [4],

Table 1. Key scheduling of GOST.

Round	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Key	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7
Round	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Key	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_7	K_6	K_5	K_4	K_3	K_2	K_1	K_0

and the selection of the S-boxes was part of the secret key. There are $2^8 \cdot 16!$ possible sets of such S-boxes [21] and thus the theoretical security level of GOST would be $256+log_2(2^8 \cdot 16!) = 256+354 = 610$ bits. However, Saarinen has shown in [21] that a chosen key attack can reveal the set of S-boxes with a complexity of only 2^{32} encryptions.

Schneier states that the Central Bank of Russian Federation has been using the S-boxes described in Table 2. This set of S-boxes serves as *one* example of GOST, but according to the standard the appropriate choice of S-boxes is a design decision. It is clear that the selection of the S-boxes has a significant influence on the cryptographic strength of the cipher, thus a careful selection is crucial. Please note that the standard does neither specify if the S-boxes used shall be different. Thus, with a small area footprint in mind, we opt for selecting one S-box that is used eight times in parallel – a similar approach as used in DESL/DESXL [14]. While DESL/DESXL lead to a slightly modified standard algorithm, in the case of GOST we end up with a solution that is even conform to the standard. As Biham *et al.* have pointed out in [1], the S-boxes do not even have to be permutations. Hence theoretically, it would be even possible to use simple wiring, which further decreases the area footprint. Of course the differential and linear properties of such an implementation will be very weak and thus this is not an interesting option.

Table 2. Set of GOST S-boxes as used by the Central Bank of Russian Federation [22].

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_1(x)$	4	A	9	2	D	8	0	E	6	B	1	C	7	F	5	3
$S_2(x)$	E	B	4	C	6	D	F	A	2	3	8	1	0	7	5	9
$S_3(x)$	5	8	1	D	A	3	4	2	E	F	C	7	6	0	9	B
$S_4(x)$	7	D	A	1	0	8	9	F	E	4	6	C	B	2	5	3
$S_5(x)$	6	C	7	1	5	F	D	8	4	A	9	E	0	3	B	2
$S_6(x)$	4	B	A	0	7	2	1	D	3	6	8	5	9	C	F	E
$S_7(x)$	D	B	4	1	3	F	5	9	0	A	E	7	6	8	2	C
$S_8(x)$	1	F	D	0	5	7	A	4	9	2	3	E	6	B	8	C

Instead we focus on the linear and the differential properties and use the classification of 4-bit S-boxes published in [15] as a guideline for the selection of an appropriate S-box. In fact we have chosen to use the PRESENT S-box, since it is strong with regard to linear and differential properties and has the lowest

area footprint of 4-bit S-boxes [2]. Let us denote the *Fourier* coefficient of S by

$$S_b^W(a) = \sum_{x \in F_2^4} (-1)^{<b,S(x)>+<a,x>}.$$

Further, we denote a fixed non-zero input difference with $\Delta_I \in \mathbb{F}_2^4$ and a fixed non-zero output difference with $\Delta_O \in \mathbb{F}_2^4$. The design criteria of the PRESENT S-box are [2]:

1. For any fixed non-zero input difference $\Delta_I \in \mathbb{F}_2^4$ and any fixed non-zero output difference $\Delta_O \in \mathbb{F}_2^4$ we require
$$\#\{x \in \mathbb{F}_2^4 \,|S(x) + S(x + \Delta_I) = \Delta_O\} \leq 4.$$
2. For any fixed non-zero input difference $\Delta_I \in \mathbb{F}_2^4$ and any fixed output difference $\Delta_O \in \mathbb{F}_2^4$ such that $\mathrm{wt}(\Delta_I) = \mathrm{wt}(\Delta_O) = 1$ we have
$$\{x \in \mathbb{F}_2^4 \,|S(x) + S(x + \Delta_I) = \Delta_O\} = \emptyset.$$
3. For all non-zero $a \in \mathbb{F}_2^4$ and all non-zero $b \in \mathbb{F}_2^4$ it holds that $|S_b^W(a)| \leq 8$.
4. For all $a \in \mathbb{F}_2^4$ and all non-zero $b \in \mathbb{F}_2^4$ such that $\mathrm{wt}(a) = \mathrm{wt}(b) = 1$ it holds that $|S_b^W(a)| \leq 4$.

We have calculated the differential and linear distribution tables of the S-boxes used by the Central Bank of Russian Federation (S_1 to S_8) and PRESENT (S_{PS}) and list them in the appendix. From Table 3, where we summarize the linear and differential characteristics of these S-boxes, it becomes clear that the PRESENT S-box is stronger both with regard to linear and differential cryptanalysis due to the strict design criteria. Therefore in the following we will also consider a GOST implementation that uses eight times the PRESENT S-box. We will refer to this variant with the term GOST-PS while GOST-FB denotes the GOST variant that uses the S-boxes as used by the Central Bank of Russian Federation.

Table 3. Linear and Differential characteristics of GOST and PRESENT S-boxes.

Sbox	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_{PS}
max S_b^W	8	12	12	12	12	12	12	12	8
max DC	6	6	6	6	4	6	8	8	4

4 Hardware Implementations

A round-based implementation of GOST can be done straight forwardly, while a serialized implementation poses some challenges for a hardware designer. Thus we spare the details of the former architecture and focus on the latter with

a data path width of 4 bits. Most challenging is the permutation step, since it rotates by 11 bit positions. Thus it is not possible to operate on 4 bit chunks, but instead we have to operate on the whole state. In our architecture (see Figure 2) it takes 8 clock cycles to process all chunks of the state and to perform one round of GOST. Then we swap the content of the registers as it is required by the Feistel structure within one clock cycle, i.e. we operate on the whole state. We use this clock cycle to perform the 11 bit rotation, but in our architecture we have XORed both halves of the state already. Thus we have to shift the right halve in the previous clock cycles by 11 bit positions to the right, before storing it as the new left halve. Then when the XOR sum of both halves is rotated by 11 bit positions to the left, the final step of one round of GOST is performed. In short, the following operations are carried out when the content of the registers is swapped:

$$L_{i+1} = R_i \gg 11,$$
$$R_{i+1} = (L_i \oplus S(K_i + R_i \bmod 2^{32})) \ll 11.$$

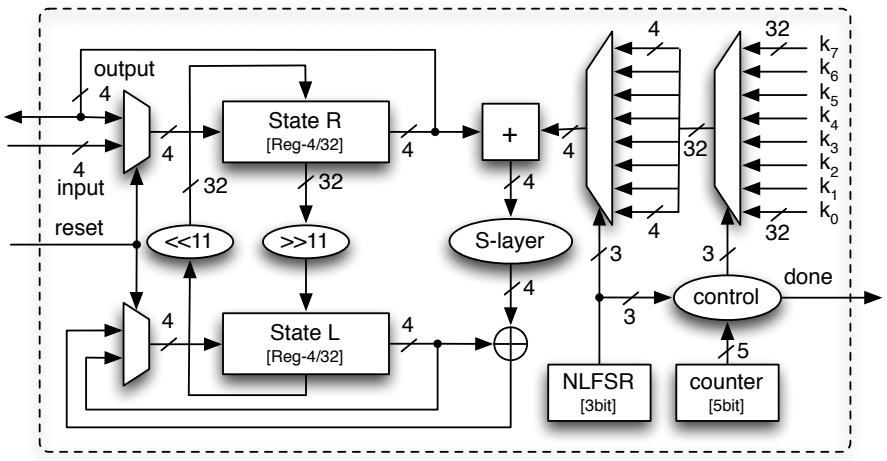

Fig. 2. Architecture of a lightweight hardware architecture with a 4-bit data path for GOST.

Recall that the key schedule of GOST is very simple: in every round a 32-bit chunk of the 256-bit key is used as the round key and for the last eight rounds the order is swapped (see Table 1). Thus apart from a 32-bit wide 8-to-1 MUX to select the right round key there is only very little logic required for a round-based implementation. For a serialized implementation we need an additional 4-bit wide 8-to-1 MUX to select the right chunk of the round key. In

total the key schedule sums up to only 99 GE for a serialized implementation and 50 GE for a round-based. If the application requires the key to be updated, 256 additional flip-flops are required for storage. However, especially in passive RFID-tag scenarios it is very unlikely that the key needs to be changed – as [2, 19] have pointed out before. Therefore the main target for our implementations are applications with a fixed key. Then only a small amount of (cheap) tie cells are required to hard-wire the key.

For functional simulation we used *Mentor Graphics ModelSimXE 6.4b* and *Synopsys DesignCompiler* version *A-2007.12-SP1* [24] was used to synthesize the designs to the *Virtual Silicon* (VST) standard cell library *UMCL18G212T3*, which is based on the *UMC L180 0.18µm 1P6M* logic process and has a typical voltage of 1.8 Volt [26]. Table 4 summarizes the synthesis results and compares them to a selection of other lightweight hardware implementations.

When the synthesis compiler is advised to use the *compile ultra* option, the smallest area can be achieved. In this case the compiler optimizes the whole design without taking care of the single entities, which makes it very difficult (if not impossible) to assign the area requirements to a single component. Thus we also advised the compiler to use the *compile simple* option, in which case the area is larger, but a more detailed breakdown is possible. Below we give such a breakdown of both GOST variants. Recall that GOST-FB refers to the variant that uses the set of S-boxes as used by the Central Bank of Russian Federation while GOST-PS refers to the variant that uses eight times the PRESENT S-box. It is noteworthy to highlight again that both variants are fully standard conform.

GOST		Federal Bank		PRESENT	
		serial	round	serial	round
cycles		264	32	264	32
t'put @100 KHz (Kbps.)		24.24	200	24.24	200
compile ultra	sum	800	1000	651	1017
compile simple	sum	876	1028	664	1055
sequential:	State	384	384	384	384
	Round counter	50	41	50	41
	serial counter	23	0	23	0
combinational:	MUX	15	0	15	0
	KeyAdd	41	271	41	271
	rotation	0	0	0	0
	sBoxLayer	239	187	27	214
	XOR	11	85	11	85
	key scheduling	99	50	99	50
	control	14	10	14	10

As one can see, a serialized implementation leads to smaller area requirements, mostly due to a scaled down key addition module (saves 230 GE) and XOR gates (saves 70 GE). However, the serialized architecture also introduces some area overhead, because additional MUXes are required: one for the state register (15 GE) and one to select the right chunk of the round key (49 GE). If the

GOST variant uses different S-boxes such as GOST-FB, but not GOST-PS, another MUX is required to select the correct S-box (52 GE). Furthermore, an NLFSR (23 GE) is required as a serial counter and the key addition requires a flip-flop to store the carry bit. The round counter is smaller in the round-based architectures, which we believe is because in the serialized architecture gated registers are required, but not in the round-based.

Table 4. Hardware implementation results of selected symmetric encryption algorithms. GOST-FB uses eight different S-boxes as used by the Central Bank of Russian Federation and GOST-PS uses eight times the PRESENT-S-box. Note that both variants are fully standard compliant.

Algorithm		key size	block size	cycles/ block	Throughput (@100 KHz)	Tech. [μm]	Area [GE]
Stream Ciphers							
Trivium	[8]	80	1	1	100	0.13	2,599
Grain	[8]	80	1	1	100	0.13	1,294
Block Ciphers							
KATAN32	[5]	80	32	255	12.5	0.13	802
KATAN48	[5]	80	48	255	18.8	0.13	927
KATAN64	[5]	80	64	255	25.1	0.13	1054
KTANTAN32	[5]	80	32	255	12.5	0.13	462
KTANTAN48	[5]	80	48	255	18.8	0.13	588
KTANTAN64	[5]	80	64	255	25.1	0.13	688
PRESENT	[20]	80	64	547	11.7	0.18	1,075
SEA	[16]	96	96	93	103	0.13	3,758
mCrypton	[3]	96	64	13	492.3	0.13	2,681
HIGHT	[10]	128	64	34	188	0.25	3,048
AES	[7]	128	128	1,032	12.4	0.35	3,400
AES	[9]	128	128	160	80	0.13	3,100
DESXL	[14]	184	64	144	44.4	0.18	2,168
GOST-FB		256	64	264	24.24	0.18	800
GOST-FB		256	64	32	200	0.18	1000
GOST-PS		256	64	264	24.24	0.18	651
GOST-PS		256	64	32	200	0.18	1017

The `sBoxLayer` module in the serialized GOST-PS is by far the smallest of all our variants, because we only need to implement one PRESENT S-box, while GOST-FB needs all 8 S-boxes and an additional MUX. It is also interesting to see that the S-boxes of GOST-FB require less area than the PRESENT S-box. We believe that this is closely related to the fact that these S-boxes are weaker with regard to differential and linear cryptanalysis (see Table 3 and the tables in the appendix).

We used *Synopsys PowerCompiler* version *A-2007.12-SP1* [25] to estimate the power consumption of our implementations. All power estimates for the smallest wire-load model (10K GE) at a supply voltage of 1.8 Volt and a frequency of 100 KHz are below 2.6 μW, which indicates that all GOST variants are well suited for demanding applications, including passive RFID tags. However, the accuracy level of simulated power figures greatly depends on the simulation tools and parameters used. Furthermore, the power consumption also strongly depends on the target library used. Thus to have a fair comparison, we do not include any power figures in Table 4.

5 Conclusions

In this paper we have revisited the former Soviet encryption standard GOST 28147-89, that has been standardized since 1989 already. Despite considerable cryptanalytic efforts spent in the past 20 years, GOST is still not broken. Since to the best of our knowledge no hardware implementations of GOST have been published so far, we have implemented GOST in hardware with a focus on a low area footprint to close this gap. As a further optimization, we exploit the fact that the standard does not specify a set of S-boxes and use a single S-box repeated eight times. We have selected the PRESENT S-box, which has the best linear and differential properties among all 4-bit S-boxes while at the same time requiring the least amount of area. Our synthesis results show that a standard conform GOST variant that uses the PRESENT S-box requires only 651 GE.

Many of the recently proposed lightweight block ciphers are not yet mature for standardization, while others, i.e. PRESENT and HIGHT, are currently undergoing standardization by ISO. At the same time, GOST is already standardized since 20 years. Furthermore, GOST offers 256 bits of security, while many lightweight proposal are limited to 128 or 80 bits. It is therefore an interesting candidate for low-cost applications that also require a very strong security level.

References

1. E. Biham, O. Dunkelman, and N. Keller. Improved slide attacks. In *Fast Software Encryption 2007 – FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 153–166. Springer, 2007.
2. A. Bogdanov, G. Leander, L. Knudsen, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT - An Ultra-Lightweight Block Cipher. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems — CHES 2007*, number 4727 in Lecture Notes in Computer Science, pages 450–466. Springer-Verlag, 2007.
3. C. Lim and T. Korkishko. mCrypton - A Lightweight Block Cipher for Security of Low-cost RFID Tags and Sensors. In J. Song, T. Kwon, and M. Yung, editors, *Workshop on Information Security Applications — WISA 2005*, volume 3786 of *Lecture Notes in Computer Science*, pages 243–258. Springer-Verlag, 2005.

4. C. Charnes, L. O'Connor, J. Pieprzyk, R. Safavi-Naini, and Y. Zheng. Further comments on the soviet encryption algorithm. In *Advances in Cryptology — EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 433–438. Springer-Verlag, 1995.
5. C. de Cannière, O. Dunkelman, and M. Knezević. Katan and ktantan–a family of small and efficient hardware-oriented block ciphers. In C. Clavier and K. Gaj, editors, *Proceedings of CHES '09*, volume 5747 of *LNCS*, pages 272–288. Springer, 2009.
6. V. Dolmatov. Gost 28147-89 encryption, decryption and mac algorithms. Available for download via `http://tools.ietf.org/html/draft-dolmatov-cryptocom-gost2814789`, December 3 2009.
7. M. Feldhofer, J. Wolkerstorfer, and V. Rijmen. AES Implementation on a Grain of Sand. *Information Security, IEE Proceedings*, 152(1):13–20, 2005.
8. T. Good and M. Benaissa. Hardware Results for Selected Stream Cipher Candidates. State of the Art of Stream Ciphers 2007 (SASC 2007), Workshop Record, February 2007. Available via `www.ecrypt.eu.org/stream`.
9. P. Hämäläinen, T. Alho, M. Hännikäinen, and T. D. Hämäläinen. Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core. In *DSD*, pages 577–583, 2006.
10. D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. S. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee. HIGHT: A New Block Cipher Suitable for Low-Resource Device. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems — CHES 2006*, number 4249 in Lecture Notes in Computer Science, pages 46–59. Springer-Verlag, 2006.
11. ISO/IEC. *International Standard ISO/IEC 18033 Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers*.
12. O. Kara. Reflection cryptanalysis of some ciphers. In *Progress in Cryptology - INDOCRYPT 2008*, volume 5365 of *Lecture Notes in Computer Science*, pages 294–307. Springer, 2008.
13. Y. Ko, S. Hong, W. L. S. Lee, and J.-S. Kang. Related Key Differential Attacks on 27 Rounds of XTEA and Full-Round GOST. In *Fast Software Encryption 2004 – FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 299–316. Springer-Verlag, 2004.
14. G. Leander, C. Paar, A. Poschmann, and K. Schramm. New Lightweight DES Variants. In *Fast Software Encryption 2007 – FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 196–210. Springer-Verlag, 2007.
15. G. Leander and A. Poschmann. On the classification of 4-Bit s-boxes. In C. Carlet and B. Sunar, editors, *Proceedings of WAIFI 2007*, volume 4547 of *LNCS*. Springer, 2007.
16. F. Mace, F.-X. Standaert, and J.-J. Quisquater. ASIC Implementations of the Block Cipher SEA for Constrained Applications. In *RFID Security — RFIDsec 2007, Workshop Record*, pages 103 – 114, Malaga, Spain, 2007.
17. National Soviet Bureau of Standards. Informtation Processing System - Cryptographic Protection - Cryptographic Algorithm GOST 28147-89, 1989.
18. G. S. Oreku, J. Li, T. Pazynyuk, and F. J. Mtenzi. Modified s-box to archive accelerated gost. *IJCSNS International Journal of Computer Science and Network Security*, 7(6):88–98, June 2007.
19. M. Robshaw. Searching for compact algorithms: CGEN. In P. Nguyen, editor, *Proceedings of Vietcrypt 2006*, volume 4341 of *LNCS*, pages 37–49. Springer-Verlag, 2006.

20. C. Rolfes, A. Poschmann, G. Leander, and C. Paar. Ultra-Lightweight Implementations for Smart Devices - Security for 1000 Gate Equivalents. In G. Grimaud and F.-X. Standaert, editors, *Smart Card Research and Advanced Application — CARDIS 2008*, volume 5189 of *Lecture Notes in Computer Science*, pages 89–103. Springer-Verlag, 2008.
21. M.-J. Saarinen. A chosen Key attack against the secret S-boxes of GOST. unpublished manuscript, 1998.
22. B. Schneier. *Applied Cryptography*. John Wiley & Sons, 2nd edition, 1996.
23. H. Seki and T. Kaneko. Differential Cryptanalysis of Reduced Rounds of GOST. In *Selected Areas in Cryptography*, volume 2012 of *Lecture Notes in Computer Science*, pages 315–323. Springer, 2001.
24. Synopsys. Design Compiler User Guide - Version A-2007.12. Available via `http://tinyurl.com/pon88o`, December 2007.
25. Synopsys. Power Compiler User Guide - Version A-2007.12. Available via `http://tinyurl.com/lfqhy5`, March 2007.
26. Virtual Silicon Inc. 0.18 μm VIP Standard Cell Library Tape Out Ready, Part Number: UMCL18G212T3, Process: UMC Logic 0.18 μm Generic II Technology: 0.18μm, July 2004.
27. M. Weiser. The computer for the 21st century. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(3):3–11, 1999.

Appendix

The following tables display the differential and linear properties of the PRESENT S-box (S_{PS}) and the GOST S-boxes as used by the Central Bank of Russian Federation (S_1 to S_8).

S_{PS}

DC	Δ_O																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	4	0	0	0	4	0	4	0	0	0	4	0	0
2	0	0	0	2	0	4	2	0	0	0	2	0	2	2	2	0
3	0	2	0	2	2	0	4	2	0	0	2	2	0	0	0	0
4	0	0	0	0	0	4	2	2	0	2	2	0	2	0	2	0
5	0	2	0	0	2	0	0	0	0	2	2	2	4	2	0	0
6	0	0	2	0	0	0	2	0	2	0	0	4	2	0	0	4
7	0	4	2	0	0	0	2	0	2	0	0	0	2	0	0	4
8	0	0	0	2	0	0	0	2	0	2	0	4	0	2	0	4
9	0	0	2	0	4	0	2	0	2	0	0	0	2	0	4	0
A	0	0	2	2	0	4	0	0	2	0	2	0	0	2	2	0
B	0	2	0	0	2	0	0	0	4	2	2	2	0	2	0	0
C	0	0	2	0	0	4	0	2	2	2	2	0	0	0	2	0
D	0	2	4	2	2	0	0	2	0	0	2	2	0	0	0	0
E	0	0	2	2	0	0	2	2	2	2	0	0	2	2	0	0
F	0	4	0	0	4	0	0	0	0	0	0	0	0	0	4	4

LC	b															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	-8	0	-8	0	0	0	0	0	-8	0	8
2	0	0	4	4	-4	-4	0	0	4	-4	0	8	0	8	-4	4
3	0	0	4	4	4	-4	-8	0	-4	4	-8	0	0	0	-4	-4
4	0	0	-4	4	-4	-4	0	8	-4	-4	0	-8	0	0	-4	4
5	0	0	-4	4	-4	4	0	0	4	4	-8	0	8	0	4	4
6	0	0	0	-8	0	0	-8	0	0	-8	0	0	8	0	0	0
7	0	0	0	8	8	0	0	0	0	-8	0	0	0	0	8	0
8	0	0	4	-4	0	0	-4	4	-4	4	0	0	-4	4	8	8
9	0	8	-4	-4	0	0	4	-4	-4	-4	-8	0	-4	4	0	0
A	0	0	8	0	4	4	4	-4	0	0	0	-8	4	4	-4	4
B	0	-8	0	0	-4	-4	4	-4	-8	0	0	0	4	4	4	-4
C	0	0	0	0	-4	-4	-4	-4	8	0	0	-8	-4	4	4	-4
D	0	8	8	0	-4	-4	4	4	0	0	0	0	4	-4	4	-4
E	0	0	4	4	-8	8	-4	-4	-4	-4	0	0	-4	-4	0	0
F	0	8	-4	4	0	0	-4	-4	-4	4	8	0	4	4	0	0

S_1

DC \ Δ_O	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	2	2	0	2	0	0	2	0	4	4	0
2	0	0	2	0	0	0	2	4	2	0	0	0	2	4	0	0
3	0	0	0	4	2	0	2	0	2	0	6	0	0	0	0	0
4	0	2	2	0	4	0	0	0	0	4	0	0	2	0	0	2
5	0	0	4	0	0	0	0	4	0	4	0	0	4	0	0	0
6	0	0	0	4	6	0	2	0	2	0	2	0	0	0	0	0
7	0	2	0	0	0	2	0	0	0	0	4	2	0	0	4	2
8	0	2	2	0	0	2	0	2	2	0	2	0	0	2	2	0
9	0	0	2	4	0	2	0	0	0	0	0	2	2	0	0	4
A	0	2	0	0	0	2	2	2	2	2	0	2	0	0	0	2
B	0	0	2	0	2	0	0	0	2	2	0	2	0	2	2	2
C	0	4	2	4	0	2	0	0	0	0	0	2	2	0	0	0
D	0	0	0	0	0	0	2	2	0	0	2	2	2	2	2	2
E	0	2	0	0	2	2	2	0	0	2	0	0	2	2	2	0
F	0	2	0	0	0	2	2	2	2	2	0	2	0	0	0	2

Δ_I

LC \ b	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	-4	8	-4	-4	0	4	0	8	4	0	4	-4	0	4	0
2	0	0	-4	4	-4	4	0	0	-4	4	0	0	-8	-8	4	-4
3	0	4	-4	0	-8	-4	-4	0	4	0	-8	4	4	0	0	-4
4	0	4	0	-4	4	-8	-4	-8	0	4	0	-4	-4	0	4	0
5	0	0	0	0	0	-8	8	0	0	0	0	0	0	-8	-8	0
6	0	4	-4	0	0	-4	-4	8	4	0	8	4	-4	0	0	4
7	0	8	4	4	-4	4	0	0	4	4	0	-8	0	0	-4	4
8	0	8	4	-4	4	4	0	0	-4	4	0	8	0	0	-4	-4
9	0	-4	4	8	0	-4	-4	0	-4	8	0	4	4	0	0	4
A	0	0	8	0	0	0	-8	0	0	-8	0	0	0	-8	0	0
B	0	-4	0	-4	-4	0	-4	0	0	4	8	-4	4	0	-4	-8
C	0	-4	-4	0	0	4	-4	-8	4	0	0	4	-4	0	-8	4
D	0	0	4	4	-4	-4	0	0	-4	-4	0	0	-8	8	-4	-4
E	0	4	0	4	-4	0	4	-8	0	-4	8	4	4	0	4	0
F	0	0	0	8	8	0	0	0	8	0	0	0	0	0	0	-8

a

S_2

DC \ Δ_O	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	2	0	0	0	4	0	2	2	2	0	2	2	0	0	0
2	0	0	2	0	0	2	0	4	0	2	4	0	0	0	2	0
3	0	0	6	2	0	0	0	0	0	2	0	2	2	0	0	2
4	0	0	2	0	2	0	4	0	4	0	0	2	0	2	0	0
5	0	2	0	6	2	2	0	0	0	0	0	0	0	2	2	0
6	0	6	2	0	0	0	2	2	2	0	2	0	0	0	0	0
7	0	2	0	0	4	0	2	0	0	2	2	2	0	0	0	2
8	0	0	0	2	0	0	2	0	2	0	4	0	4	2	0	0
9	0	2	0	0	2	2	2	0	0	2	0	0	0	4	0	2
A	0	0	0	2	2	0	4	0	0	0	2	0	0	2	0	4
B	0	0	0	2	0	0	0	2	4	0	2	0	0	0	2	4
C	0	2	0	0	2	2	0	2	0	0	0	2	2	0	4	0
D	0	0	2	0	0	2	0	0	0	4	0	2	0	2	2	2
E	0	0	2	0	2	0	0	0	0	2	0	4	2	2	2	0
F	0	0	0	2	0	2	0	4	2	0	0	0	4	0	2	0

Δ_I

LC \ b	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	8	0	0	-4	4	4	4	4	4	4	-4	0	0	-8	0
2	0	0	-8	0	0	0	0	8	4	4	-4	4	4	4	4	-4
3	0	8	0	-8	4	-4	4	-4	0	0	0	0	4	4	4	4
4	0	4	0	4	4	-8	-4	0	0	-4	0	-4	4	0	-4	-8
5	0	4	0	-4	0	4	0	-4	-4	0	-4	8	-4	0	-4	-8
6	0	-4	-8	-4	4	0	-4	0	-4	0	4	0	0	4	-8	4
7	0	-4	0	-4	-8	4	0	-4	0	-4	0	-4	8	4	0	-4
8	0	4	-4	0	-8	-4	-4	0	-8	4	4	0	0	-4	4	0
9	0	-4	-4	0	-4	-8	8	-4	4	0	0	4	0	-4	-4	0
A	0	-4	4	-8	0	-4	4	8	-4	0	0	-4	-4	0	0	-4
B	0	4	-4	0	-4	0	0	4	0	-12	-4	0	-4	0	0	4
C	0	0	4	4	-4	-4	0	0	0	0	4	4	-4	12	0	0
D	0	0	4	-4	0	0	-4	4	4	-4	8	8	4	-4	0	0
E	0	0	-4	4	4	4	8	0	-4	-4	8	0	0	0	4	-4
F	0	0	4	4	0	0	4	4	-8	0	-4	4	8	0	-4	4

a

S_3

DC \ Δ_O	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	2	2	0	0	0	4	0	0	2	0	2	2	2	0	0
2	0	2	2	0	2	2	0	0	2	0	0	2	0	0	2	2
3	0	0	0	2	0	0	0	2	4	6	0	0	0	2	0	0
4	0	0	0	0	0	4	0	0	2	0	0	2	2	0	0	6
5	0	0	2	2	0	0	2	2	0	4	0	0	0	0	4	0
6	0	2	0	0	2	0	0	4	0	0	4	2	0	0	2	0
7	0	2	2	0	0	2	2	4	0	0	0	0	4	0	0	0
8	0	0	0	2	0	0	0	2	0	2	2	2	2	4	0	0
9	0	2	0	0	0	2	4	0	0	0	4	2	0	0	0	2
A	0	0	6	2	0	0	0	0	2	2	0	0	0	0	0	4
B	0	2	2	2	6	0	0	0	0	0	2	0	0	0	2	0
C	0	0	0	2	2	2	2	0	6	0	0	0	2	0	0	0
D	0	0	0	2	2	4	0	0	0	0	2	0	0	2	4	0
E	0	0	0	2	2	0	2	2	0	0	2	0	2	4	0	0
F	0	4	0	0	0	0	0	0	0	0	0	4	2	2	2	2

Δ_I

LC \ b	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	4	4	-8	-4	-8	0	-4	0	-4	4	0	-4	0	0	4
2	0	4	-4	0	0	4	4	8	0	-4	4	0	0	-4	-4	8
3	0	0	-8	-8	-4	4	-4	4	0	0	0	0	-4	4	4	-4
4	0	-4	4	0	-8	4	4	0	-4	-8	0	-4	4	0	0	-4
5	0	0	0	0	4	-4	-4	4	4	-4	4	-4	8	8	0	0
6	0	0	0	0	0	0	0	0	-4	4	4	-4	-4	4	-12	-4
7	0	-4	4	0	-4	0	0	4	4	0	0	12	0	4	-4	0
8	0	0	4	-4	4	4	-8	0	4	-4	0	0	0	-8	-4	-4
9	0	-4	0	4	0	4	0	-4	4	0	12	0	-4	0	4	0
A	0	-4	0	4	4	0	-4	0	-4	-8	-4	0	-8	4	0	4
B	0	0	4	-4	0	8	-4	-4	-4	4	0	0	4	4	0	8
C	0	-4	8	-4	4	0	4	8	0	4	0	-4	-4	0	4	0
D	0	-8	-4	-4	0	0	4	-4	8	0	-4	-4	0	0	-4	4
E	0	8	4	4	-4	4	0	0	8	0	-4	-4	-4	4	0	0
F	0	-4	0	4	-8	-4	-8	4	0	4	0	-4	0	-4	0	4

a

S_4

DC \ Δ_O	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	4	0	2	2	6	2	0	0	0	0
2	0	2	0	0	0	0	0	2	4	2	0	0	2	2	2	0
3	0	2	4	0	0	0	2	4	2	0	0	0	0	0	0	2
4	0	0	0	4	0	4	2	2	0	0	0	0	0	0	2	2
5	0	0	0	0	0	4	0	0	2	2	0	0	2	2	0	4
6	0	0	2	0	0	0	0	2	0	2	2	2	0	2	4	0
7	0	4	2	0	4	0	0	2	2	0	0	0	0	2	0	0
8	0	0	0	0	0	0	0	0	0	4	2	2	6	2	0	0
9	0	0	2	6	0	0	2	2	0	0	4	0	0	0	0	0
A	0	4	2	0	2	4	0	0	0	0	0	2	0	2	0	0
B	0	0	0	2	2	0	0	0	0	0	0	6	0	2	2	2
C	0	0	2	2	0	0	0	0	0	0	0	0	4	0	2	6
D	0	0	0	0	4	4	4	0	0	4	0	0	0	0	0	0
E	0	2	2	2	2	0	2	2	0	0	0	2	0	0	2	0
F	0	2	0	0	2	0	0	0	4	0	2	0	2	2	2	0

Δ_I

LC \ b	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	-4	-4	0	0	4	4	0	0	4	4	-8	8	4	4
2	0	4	0	4	0	4	0	4	0	4	0	4	0	4	0	-12
3	0	-4	-4	8	0	-4	4	0	0	-4	4	0	8	4	4	0
4	0	4	0	-4	-8	-4	0	-4	0	-4	0	4	0	-4	8	-4
5	0	4	-12	0	0	4	4	0	0	-4	-4	0	0	-4	-4	0
6	0	-8	0	0	-8	0	0	0	0	0	0	8	0	0	-8	0
7	0	0	4	-4	0	0	4	-4	0	-8	-4	-4	0	8	-4	-4
8	0	-4	4	0	4	8	8	-4	-4	0	0	4	0	-4	4	0
9	0	4	0	4	4	0	-4	-8	4	0	-4	8	0	4	0	4
A	0	0	4	4	-4	4	0	8	4	-4	-8	0	0	0	4	4
B	0	0	0	0	-4	4	4	-4	12	4	4	-4	0	0	0	0
C	0	-8	-4	-4	4	4	-8	0	4	-4	0	0	0	0	4	-4
D	0	0	0	8	-4	4	-4	-4	-4	-4	4	-4	-8	0	0	0
E	0	-4	-4	0	-4	0	0	-4	-4	8	-8	-4	0	4	4	0
F	0	-4	0	4	4	-8	4	0	4	0	-4	0	-8	-4	0	-4

a

S_5

DC \ Δ_O	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	2	2	2	0	2	4	0	0	0	2	0
2	0	4	0	0	2	0	0	2	2	0	0	2	0	4	0	0
3	0	0	4	2	0	0	0	2	2	0	2	2	0	2	0	0
4	0	0	2	4	2	0	0	0	0	4	2	0	2	0	0	0
5	0	0	0	0	0	2	0	2	0	4	2	2	2	0	0	2
6	0	0	2	0	2	0	0	0	2	2	0	2	0	2	2	2
7	0	4	0	0	2	0	2	0	2	0	2	0	0	0	4	0
8	0	0	2	0	0	2	4	0	0	0	2	0	2	0	2	2
9	0	0	0	2	0	0	2	0	4	2	0	0	2	0	0	4
A	0	0	2	2	0	0	0	0	0	0	0	4	0	4	2	2
B	0	0	0	0	2	4	0	2	4	0	0	0	0	2	2	0
C	0	2	0	2	2	2	4	0	0	0	0	0	2	0	0	2
D	0	2	0	2	0	4	0	0	0	0	2	2	2	0	0	2
E	0	2	4	0	0	0	0	2	0	2	0	0	2	2	2	0
F	0	2	0	0	4	0	2	4	0	0	0	2	2	0	0	0

Δ_I

LC \ b	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	-4	4	0	-4	0	0	4	4	8	0	4	-8	4	4	0
2	0	4	0	4	-4	0	-4	0	4	0	4	0	0	-4	0	12
3	0	8	4	-4	0	0	4	4	8	0	4	-4	0	0	-4	-4
4	0	4	0	-4	-4	-8	4	-8	0	-4	0	4	-4	0	4	0
5	0	0	-4	4	0	0	4	-4	4	4	-8	0	-4	-4	-8	0
6	0	8	0	0	0	0	-8	0	-4	4	-4	-4	-4	-4	4	-4
7	0	-4	-4	0	-4	-8	0	4	0	4	4	0	4	-8	0	-4
8	0	-4	4	0	-8	4	4	0	0	-4	-4	-8	0	-4	4	0
9	0	0	-8	-8	-4	4	-4	4	4	-4	-4	4	0	0	0	0
A	0	0	-4	-4	-4	-4	0	0	-4	4	0	-8	0	8	-4	4
B	0	-4	0	-4	8	-4	0	4	0	-4	0	-4	-8	-4	0	4
C	0	0	-4	4	4	-4	0	0	8	0	-4	-4	4	4	8	0
D	0	4	-8	4	0	4	8	4	-4	0	4	0	-4	0	4	0
E	0	4	4	0	0	-4	4	8	-4	0	-8	4	4	0	0	4
F	0	0	0	-8	4	4	4	-4	0	8	0	0	4	-4	4	4

a

S_6

DC \ Δ_O	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	2	0	0	0	6	0	0	0	0	2	0	2	2	0	2
2	0	0	2	2	0	0	4	0	0	0	0	4	0	0	2	2
3	0	2	0	4	2	0	2	2	0	0	2	0	0	0	2	0
4	0	0	0	2	0	0	0	2	0	2	4	4	0	2	0	0
5	0	2	0	0	0	0	4	2	0	0	2	0	2	0	0	4
6	0	2	2	0	0	2	2	0	2	2	0	0	2	2	0	0
7	0	0	0	0	2	0	0	2	2	4	2	0	2	2	0	0
8	0	0	2	2	0	2	0	2	0	0	0	0	0	2	6	0
9	0	0	4	0	0	0	0	0	4	0	0	4	0	0	0	4
A	0	2	0	0	0	0	2	0	4	2	0	0	4	0	2	0
B	0	2	0	4	2	0	0	0	0	2	0	0	2	4	0	0
C	0	0	0	0	4	2	0	2	2	2	0	0	0	2	2	0
D	0	4	2	0	4	2	0	0	2	0	0	0	0	0	0	2
E	0	0	2	2	0	2	0	2	0	0	0	4	2	0	0	2
F	0	0	2	0	2	0	2	2	0	2	4	0	0	0	2	0

Δ_I

LC \ b	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	-4	0	4	4	0	4	-8	0	-4	0	4	-4	-8	-4	0
2	0	0	-4	4	0	-8	-4	-4	4	4	-8	0	-4	4	0	0
3	0	-4	4	0	-4	0	0	4	4	0	0	12	0	4	-4	0
4	0	4	0	4	4	0	-4	8	4	0	4	0	-8	-4	0	4
5	0	8	0	0	0	0	-8	0	-4	-4	-4	4	4	-4	-4	-4
6	0	-4	-4	0	-4	-8	0	4	0	4	4	0	4	-8	0	-4
7	0	0	4	4	0	0	-4	-4	8	0	4	-4	8	0	-4	4
8	0	0	0	0	4	4	-4	-4	4	4	4	4	0	0	8	-8
9	0	4	0	-4	-8	-4	0	-4	4	-8	4	0	-4	0	4	0
A	0	0	-4	4	-4	4	0	0	0	0	4	-4	-4	4	-8	-8
B	0	4	4	8	-8	4	4	0	0	4	-4	0	0	-4	4	0
C	0	4	0	4	0	-4	0	-4	-8	4	8	4	0	4	0	4
D	0	0	0	-8	-4	4	-4	-4	0	8	0	0	-4	-4	-4	4
E	0	-4	12	0	0	-4	-4	0	-4	0	0	-4	-4	0	0	-4
F	0	8	4	-4	4	-4	8	0	4	4	0	0	0	0	-4	-4

a

S_7

DC \ Δ_O	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	2	2	0	0	2	2	0	4	0	4	0
2	0	0	0	0	4	0	4	0	0	2	2	0	0	2	2	0
3	0	0	0	0	2	0	0	2	0	0	8	0	2	0	0	2
4	0	2	2	0	2	0	2	0	2	0	0	2	2	0	2	0
5	0	0	4	0	2	2	0	0	4	0	0	0	2	2	0	0
6	0	0	4	0	0	0	2	2	4	0	0	0	0	0	2	2
7	0	2	2	0	2	0	2	0	2	0	0	2	2	0	2	0
8	0	2	0	0	0	4	2	4	0	0	2	0	0	2	0	0
9	0	0	0	2	0	0	0	2	0	4	0	6	0	0	0	2
A	0	4	0	6	2	0	0	0	0	0	0	2	2	0	0	0
B	0	2	0	0	0	2	0	0	0	0	2	0	0	4	2	4
C	0	0	0	4	0	2	2	0	0	0	0	4	0	2	2	0
D	0	0	2	2	0	2	0	2	2	2	0	0	0	2	0	2
E	0	0	2	2	0	2	0	2	2	2	0	0	0	2	0	2
F	0	4	0	0	2	0	0	2	0	4	0	0	2	0	0	2

Δ_I

LC \ b	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	4	0	-4	-4	0	-4	8	8	4	0	4	4	0	-4	0
2	0	0	-4	4	4	4	0	8	-4	4	0	0	0	-8	4	4
3	0	-4	4	0	0	-4	4	0	4	0	-8	4	4	0	8	4
4	0	0	0	0	0	0	0	0	0	0	8	8	0	0	8	-8
5	0	4	8	4	-4	0	4	0	-8	4	0	4	4	0	-4	0
6	0	0	4	-4	4	4	-8	0	-4	4	0	0	0	8	4	4
7	0	-4	4	0	0	12	4	0	4	0	0	-4	4	0	0	-4
8	0	-12	4	0	0	-4	-4	0	0	4	4	0	0	-4	-4	0
9	0	0	4	4	-4	4	-8	0	0	-8	-4	4	-4	-4	0	0
A	0	-4	-8	4	-4	0	-4	0	-4	0	-4	0	8	4	0	-4
B	0	0	0	-8	8	0	0	0	-4	-4	-4	4	4	-4	-4	-4
C	0	4	4	0	0	-4	-4	0	0	4	-4	-8	0	-4	4	-8
D	0	0	-4	-4	-4	4	0	-8	0	8	-4	4	-4	-4	0	0
E	0	-4	0	-4	-4	0	4	8	-4	0	-4	0	-8	4	0	-4
F	0	0	0	-8	-8	0	0	0	-4	-4	4	-4	4	-4	4	4

a

S_8

DC \ Δ_O	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	2	0	2	0	0	0	0	0	0	2	0	6	4	0
2	0	0	0	2	0	0	0	2	0	0	2	0	4	0	2	4
3	0	6	2	2	0	0	0	2	0	0	2	0	0	2	0	0
4	0	0	2	0	4	0	0	2	2	2	0	2	0	0	0	2
5	0	0	2	0	2	0	4	0	0	2	4	0	0	0	0	2
6	0	2	0	0	0	4	0	2	2	0	0	4	0	0	2	0
7	0	0	0	0	0	8	0	0	4	0	4	0	0	0	0	0
8	0	0	2	2	0	0	0	0	4	0	0	0	2	2	4	0
9	0	2	0	6	0	0	4	0	0	0	0	0	2	0	2	0
A	0	2	4	0	2	0	0	0	0	0	0	2	2	2	0	2
B	0	2	2	0	0	0	0	0	0	4	0	0	2	0	0	6
C	0	0	0	0	2	4	0	2	0	2	2	0	4	0	0	0
D	0	2	0	0	2	0	0	4	2	2	2	0	0	0	2	0
E	0	0	0	4	0	0	4	0	0	4	0	4	0	0	0	0
F	0	0	0	0	2	0	4	2	2	0	0	2	0	4	0	0

Δ_I

LC \ b	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	-4	4	-8	4	0	0	4	0	-4	-4	0	-4	-8	0	4
2	0	-8	-4	-4	0	0	4	-4	4	4	0	-8	-4	4	0	0
3	0	4	8	-4	-4	-8	4	0	4	0	4	0	0	4	0	4
4	0	-4	0	-4	4	0	4	0	0	4	0	4	12	0	-4	0
5	0	-8	4	4	0	-8	-4	-4	0	0	-4	4	0	0	4	-4
6	0	4	4	0	4	0	0	-12	-4	0	0	-4	0	-4	-4	0
7	0	0	0	0	8	0	8	0	-4	-4	4	4	-4	4	4	-4
8	0	-4	4	0	-4	8	0	-4	4	0	8	4	0	-4	4	0
9	0	0	0	-8	0	0	-8	0	-4	-4	4	-4	4	4	4	-4
A	0	-4	0	4	-4	0	4	0	0	-12	0	-4	4	0	-4	0
B	0	0	4	4	8	0	-4	4	8	0	4	-4	0	0	-4	-4
C	0	0	4	4	0	0	4	4	-4	4	0	-8	4	-4	8	0
D	0	4	0	-4	-4	0	4	0	4	0	-4	0	0	-4	0	-12
E	0	0	-8	0	0	-8	0	0	0	0	8	0	0	-8	0	0
F	0	4	-4	0	4	0	0	-4	8	-4	-4	0	4	0	8	4

a