Piccolo: An Ultra-Lightweight Blockcipher

Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai

Sony Corporation
1-7-1 Konan, Minato-ku, Tokyo 108-0075, Japan
{Kyoji.Shibutani,Takanori.Isobe,Harunaga.Hiwatari,Atsushi.Mitsuda, Toru.Akishita,Taizo.Shirai}@jp.sony.com

Abstract. We propose a new 64-bit blockcipher *Piccolo* supporting 80 and 128-bit keys. Adopting several novel design and implementation techniques, *Piccolo* achieves both high security and notably compact implementation in hardware. We show that *Piccolo* offers a sufficient security level against known analyses including recent related-key differential attacks and meet-in-the-middle attacks. In our smallest implementation, the hardware requirements for the 80 and the 128-bit key mode are only 683 and 758 gate equivalents, respectively. Moreover, *Piccolo* requires only 60 additional gate equivalents to support the decryption function due to its involution structure. Furthermore, its efficiency on the energy consumption which is evaluated by energy per bit is also remarkable. Thus, *Piccolo* is one of the competitive ultra-lightweight blockciphers which are suitable for extremely constrained environments such as RFID tags and sensor nodes.

Keywords: blockcipher, generalized Feistel networks, related-key differential attacks, meet-in-the-middle attacks, ultra-lightweight.

1 Introduction

Background and Motivation. Blockciphers are essential primitives for cryptographic applications such as data integrity, confidentiality, and protection of privacy. At the same time, with the large deployment of low resource devices such as RFID tags and sensor nodes and increasing need to provide security among such devices, lightweight cryptography has become a hot topic. Hence, recently, research on designing and analyzing lightweight blockciphers has received a lot of attention. In fact, there have been several blockciphers designed for a lightweight hardware implementation such as mCrypton [28], HIGHT [20], DESL/DESXL [27], PRESENT [11], KATAN/KTANTAN [13] and PRINTcipher [25]. The structures of these ciphers are generally categorized into two structures: Substitution Permutation Networks (SPNs) and Feistel-type structures[1].

SPNs are known as the basic structure of the current U.S. encryption standard AES [16]. Also, several lightweight blockciphers based on an SPN have been

[1] KATAN/KTANTAN is exceptional, which is based on a stream cipher.

B. Preneel and T. Takagi (Eds.): CHES 2011, LNCS 6917, pp. 342–357, 2011.

published. PRESENT consisting of an SPN is supposed to be competitive ciphers among them, since its required gate is comparable with compact stream ciphers such as Grain and Trivium[2] [19,15]. Recently, PRINTcipher was designed for IC-printing, which is also an instantiation of an SPN. It achieves remarkably compact implementation, though it has uncommon block size, i.e., 48 or 96 bits. mCrypton, which is a miniature of Crypton [29], also adopts an SPN.

On the other hand, Feistel-type structures including Feistel networks and generalized Feistel networks (GFNs) are the other most widely used structure and known as the basic structure of the former U.S. encryption standard DES [17]. Though a lot of lightweight blockciphers instantiated by the Feistel-type structure have also been published, most of them have security problems in contrast to the SPN based designs. HIGHT was designed for low resource devices, which is a variant of GFN. While it is relatively light, it has been theoretically broken by a related-key differential attack [26]. GOST is known as the former Soviet encryption standard, and has Feistel network [32]. Since the compact implementation result on GOST requiring 651 GE has been published [35], it is considered as one of the ultra-lightweight blockciphers. However it has also been theoretically broken by an improved three-subset meet-in-the-middle (MITM) attack [21].

These attacks basically rely on the slow diffusion of the Feistel-type structures and high controllability of round keys caused by a simple key schedule. Thus, to avoid those attacks, the Feistel-type structures generally require a larger number of rounds than an SPN based construction. Since this reduces the efficiency on the energy consumption, the Feistel-type structure does not seem to be suitable for lightweight blockciphers. However, it has a lot of distinct features from those of SPNs. For instance, the Feistel-type structure has a smaller round function than SPNs, since only half of the data are updated per one round. Moreover the Feistel-type structure can support a decryption function without much implementation cost. As discussed in [11], by using the counter-mode, any encryption-only ciphers can support decryption function. Yet, if the cipher itself supports decryption function, it can be used for more applications, e.g., an application requiring CBC-mode. Also, a diversity of designs is considered to be important. Thus, it is meaningful to think about design possibilities of a Feistel-type structure based lightweight blockcipher that is not only efficient but also secure against known attacks including the above explained powerful attacks.

Efficiency Metrics. While hardware efficiency can be measured in many different ways, both the *energy* consumption and the *power* consumption are important measure for lightweight applications. The energy consumption is considered as a metric for active devices which have an own power supply, and the power consumption for passive devices which do not have an own power supply. Though the power consumption heavily depends on the used technology and the EDA tool, it is well known that it is proportional to the area requirement at low frequencies, e.g., 100 kHz [25]. Thus, we adopt the area requirement, i.e., *gate equivalents* (GE) as the measure to evaluate the efficiency with respect to

[2] Note that the expected security of them against distinguish attacks is substantially higher than that of 64-bit lightweight blockciphers.

Table 1. Comparative results in hardware implementations

Algorithm	block size [bit]	key size [bit]	type	serialized arch. area [GE]	serialized arch. cycles/ block	round-based arch. area [GE]	round-based arch. cycles/ block	round-based arch. energy/^{*1} bit	round-based arch. FOM^{*2}
DESXL [27]	64	184	Feistel	2,168	144	-	-	-	-
†HIGHT [20]*	64	128	GFN	-	-	3,048	34	1,620	202
mCrypton-96 [28]	64	96	SPN	-	-	2,681	13	545	684
mCrypton-128 [28]	64	128	SPN	-	-	2,949	13	600	566
PRESENT-80 [36,11]	64	80	SPN	1,000	547	1,570	32	785	811
KATAN64 [13]	64	80	stream	1,054	254	-	-	-	-
‡KTANTAN64 [13]	64	80	stream	688	254	-	-	-	-
‡GOST-PS [35]	64	256	Feistel	651	264	1,017	32	509	1,933
‡GOST-FB [35]	64	256	Feistel	800	264	1,000	32	500	2,000
Piccolo-80	64	80	GFN	**683**	432	**1,136**	27	**480**	**1,836**
Piccolo-128	64	128	GFN	**758**	528	**1,197**	33	**618**	**1,353**
Piccolo-80*	64	80	GFN	**743**	432	**1,274**	27	**538**	**1,460**
Piccolo-128*	64	128	GFN	**818**	528	**1,362**	33	**703**	**1,045**
AES-128 [31],[38]*	128	128	SPN	2,400	226	12, 454^{*3}	11	1,071	75
CLEFIA-128 [1],[40]*	128	128	GFN	2,488	328	5,979	18	841	202
PRINTcipher-48 [25]	48	80	SPN	402	768	503	48	503	3,952
PRINTcipher-96 [25]	96	160	SPN	726	3,072	967	96	967	1,069

†: Theoretically broken under related-key setting [26].
‡: Theoretically broken under single-key setting [12,21].
$\star$: Including decryption function. The others support encryption-mode only.
$*1$: energy / bit = (area [GE] $\times$ required cycles for one block process [cycle]) / block size [bit].
$*2$: FOM = (nanobit per cycles) / area squared [GE^2].
$*3$: This implementation is not intended to be high efficiency but high throughput.

the power consumption in this work. The energy consumption is the power consumption over a certain time period, and for one block process, it is evaluated by multiplying the area requirements with the required cycles for one block. Then, by dividing the power estimation for one block process by the block size, we obtain *energy per bit* as the fair measure for the energy consumption. *FOM* (in nano bits per clock cycle per GE squared) proposed by [4] is known as another metric for energy consumption. In this work, we mainly adopt the above mentioned measures *area requirement*, *energy per bit* and *FOM* for the efficiency comparison.

Contributions and Outline. In this paper, we propose a new lightweight blockcipher *Piccolo* which is optimized for extremely constrained devices. *Piccolo* supports 64-bit block with 80 or 128-bit keys, and has an iterative structure which is a variant of a generalized Feistel network. We demonstrate that *Piccolo* offers a sufficient security level against known analyses including recent related-key differential and MITM attacks. Moreover, we present that *Piccolo* achieves remarkably compact implementation in hardware. In our smallest implementation, the area requirements for the 80 and the 128-bit key mode are only 683 and 758 GE with 432 and 528 cycles per block, respectively. The efficiency on the energy consumption evaluated by energy per bit is 480 for the 80-bit key mode, which is the smallest class among current lightweight blockciphers in literature. Furthermore, *Piccolo* requires only 60 additional GE to support decryption function. Therefore, *Piccolo* supporting both encryption and decryption functions is still comparable to other encryption-only lightweight blockciphers. These

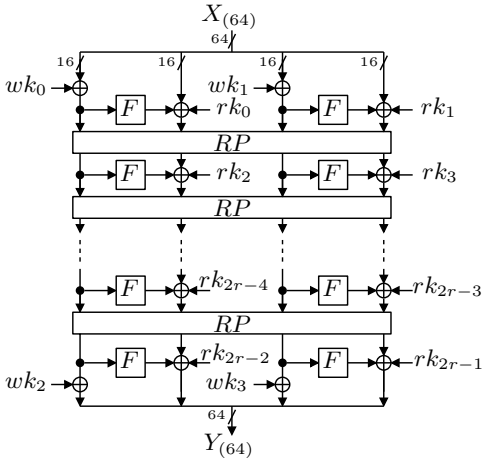

Fig. 1. Encryption function G_r

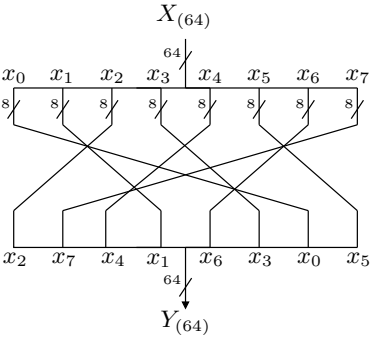

Fig. 2. Round permutation RP

comparative results regarding the hardware efficiency for lightweight blockciphers whose key size is more than 80 bits are summarized in Table 1. Note that, in our implementations, a key input is assumed to hold its value during the block process. Thus, *Piccolo* achieves both high security and extremely compact implementation unlike the other Feistel-type structure based lightweight blockciphers.

This paper is organized as follows. The specification of *Piccolo* is given in Section 2. Section 3 describes the design rationale. Sections 4 and 5 provide results on security and hardware implementation, respectively. Finally, we conclude in Section 6.

2 Specification

This section provides the specification of *Piccolo*. *Piccolo* is a 64-bit blockcipher supporting 80 and 128-bit keys. The 80 and the 128-bit key mode are referred as *Piccolo*-80 and *Piccolo*-128, respectively. Both ciphers consist of a data processing part and a key scheduling part. The differences between two key modes lie in the number of rounds for the data processing part and the key scheduling part. We first give notations used throughout this paper, then define each part.

2.1 Notations

$a_{(b)}$	: b denotes the bit length of a.
$a\|b$ or $(a\|b)$	: Concatenation.
$a \leftarrow b$	: Updating a value of a by a value of b.
${}^t\boldsymbol{a}$	: Transposition of a vector or a matrix $\boldsymbol{a}$.
$\{a\}_b$	: Representation in base b.

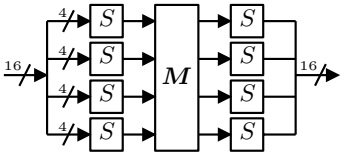

Fig. 3. F-function

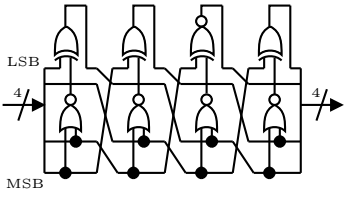

Fig. 4. S-box

2.2 Data Processing Part

The data processing part of *Piccolo* consisting of r rounds, G_r, takes a 64-bit data $X \in \{0,1\}^{64}$, four 16-bit whitening keys $wk_i \in \{0,1\}^{16} (0 \leq i < 4)$ and $2r$ 16-bit round keys $rk_i \in \{0,1\}^{16} (0 \leq i < 2r)$ as the inputs, and outputs a 64-bit data $Y \in \{0,1\}^{64}$. G_r is defined as follows:

$$G_r : \begin{cases} \{0,1\}^{64} \times \{\{0,1\}^{16}\}^4 \times \{\{0,1\}^{16}\}^{2r} \rightarrow \{0,1\}^{64} \\ (X_{(64)}, wk_{0(16)}, ..., wk_{3(16)}, rk_{0(16)}, ..., rk_{2r-1(16)}) \mapsto Y_{(64)} \end{cases}$$

Algorithm $G_r(X_{(64)}, wk_0, ..., wk_3, rk_0, ..., rk_{2r-1})$ **:**
$X_{0(16)}|X_{1(16)}|X_{2(16)}|X_{3(16)} \leftarrow X_{(64)}$
$X_0 \leftarrow X_0 \oplus wk_0$, $X_2 \leftarrow X_2 \oplus wk_1$
for $i \leftarrow 0$ to $r-2$ do
  $X_1 \leftarrow X_1 \oplus F(X_0) \oplus rk_{2i}$, $X_3 \leftarrow X_3 \oplus F(X_2) \oplus rk_{2i+1}$
  $X_0|X_1|X_2|X_3 \leftarrow RP(X_0|X_1|X_2|X_3)$
$X_1 \leftarrow X_1 \oplus F(X_0) \oplus rk_{2r-2}$, $X_3 \leftarrow X_3 \oplus F(X_2) \oplus rk_{2r-1}$
$X_0 \leftarrow X_0 \oplus wk_2$, $X_2 \leftarrow X_2 \oplus wk_3$
$Y_{(64)} \leftarrow X_0|X_1|X_2|X_3$

where F is a 16-bit F-function and RP is a 64-bit permutation defined in the following sections. The decryption function G_r^{-1} is obtained from G_r by simply changing the order of whitening and round keys as follows:

$$G_r^{-1} : \begin{cases} \{0,1\}^{64} \times \{\{0,1\}^{16}\}^4 \times \{\{0,1\}^{16}\}^{2r} \rightarrow \{0,1\}^{64} \\ (Y_{(64)}, wk_{0(16)}, ..., wk_{3(16)}, rk_{0(16)}, ..., rk_{2r-1(16)}) \mapsto X_{(64)} \end{cases}$$

Algorithm $G_r^{-1}(Y_{(64)}, wk_0, ..., wk_3, rk_0, ..., rk_{2r-1})$ **:**
$wk'_0 \leftarrow wk_2, wk'_1 \leftarrow wk_3, wk'_2 \leftarrow wk_0, wk'_3 \leftarrow wk_1$
for $i \leftarrow 0$ to $r-1$ do
  $rk'_{2i}|rk'_{2i+1} \leftarrow \begin{cases} rk_{2r-2i-2}|rk_{2r-2i-1} \text{ (if } i \bmod 2 = 0) \\ rk_{2r-2i-1}|rk_{2r-2i-2} \text{ (if } i \bmod 2 = 1) \end{cases}$
$X_{(64)} \leftarrow G_r(Y, wk'_0, ..., wk'_3, rk'_0, ..., rk'_{2r-1})$

The number of rounds, r, is 25 and 31 for *Piccolo*-80 and -128, i.e., G_{25} and G_{31} for *Piccolo*-80 and -128, respectively (See Fig. 1).

Table 2. 4-bit bijective S-box S in hexadecimal form

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S[x]$	e	4	b	2	3	8	0	9	1	a	7	f	6	c	5	d

F-Function. F-function $F : \{0,1\}^{16} \rightarrow \{0,1\}^{16}$ consists of two S-box layers separated by a diffusion matrix (See Fig. 3). The S-box layer consists of four 4-bit bijective S-boxes S given by Table 2, and updates a 16-bit data $X_{(16)}$ as follows:

$$(x_{0(4)}, x_{1(4)}, x_{2(4)}, x_{3(4)}) \leftarrow (S(x_{0(4)}), S(x_{1(4)}), S(x_{2(4)}), S(x_{3(4)})),$$

where $X_{(16)} = x_{0(4)}|x_{1(4)}|x_{2(4)}|x_{3(4)}$. The diffusion matrix $\boldsymbol{M}$ is defined as

$$\boldsymbol{M} = \begin{pmatrix} 2\,3\,1\,1 \\ 1\,2\,3\,1 \\ 1\,1\,2\,3 \\ 3\,1\,1\,2 \end{pmatrix}.$$

Then the diffusion function updates a 16-bit data $X_{(16)}$ as follows:

$${}^t(x_{0(4)}, x_{1(4)}, x_{2(4)}, x_{3(4)}) \leftarrow \boldsymbol{M} \cdot {}^t(x_{0(4)}, x_{1(4)}, x_{2(4)}, x_{3(4)}),$$

where the multiplications between matrices and vectors are performed over GF(2^4) defined by an irreducible polynomial $x^4 + x + 1$.

Round Permutation. The round permutation $RP : \{0,1\}^{64} \rightarrow \{0,1\}^{64}$ divides a 64-bit input $X_{(64)}$ into eight 8-bit data as $X_{(64)} = x_{0(8)}|x_{1(8)}|...|x_{7(8)}$, then permutes them by the following manner:

$$RP : (x_{0(8)}, x_{1(8)}, ..., x_{7(8)}) \leftarrow (x_{2(8)}, x_{7(8)}, x_{4(8)}, x_{1(8)}, x_{6(8)}, x_{3(8)}, x_{0(8)}, x_{5(8)}).$$

Finally, the round permutation concatenates $(x_{0(8)}, x_{1(8)}, ..., x_{7(8)})$ into $X_{(64)}$ (See Fig. 2).

2.3 Key Scheduling Part

The key scheduling part of *Piccolo* supports 80 and 128-bit keys, and outputs 16-bit whitening keys $wk_{i(16)} (0 \leq i < 4)$ and round keys $rk_{j(16)} (0 \leq j < 2r)$ for the data processing part. The key scheduling functions for *Piccolo*-80 and -128 are referred as KS_r^{80} and KS_r^{128}, respectively. We first define 16-bit constants con_i^{80} and con_i^{128}, then describe each key schedule.

Constant Values. The constants con_i^{80} and con_i^{128} used in KS_r^{80} and KS_r^{128}, respectively, are generated as follows:

$$\begin{cases} (con_{2i}^{80}|con_{2i+1}^{80}) \leftarrow (c_{i+1}|c_0|c_{i+1}|\{00\}_2|c_{i+1}|c_0|c_{i+1}) \oplus \{\texttt{0f1e2d3c}\}_{16}, \\ (con_{2i}^{128}|con_{2i+1}^{128}) \leftarrow (c_{i+1}|c_0|c_{i+1}|\{00\}_2|c_{i+1}|c_0|c_{i+1}) \oplus \{\texttt{6547a98b}\}_{16}, \end{cases}$$

where c_i is a 5-bit representation of i, e.g., $c_{11} = \{\texttt{01011}\}_2$.

Key Schedule for 80-Bit Key Mode (KS_r^{80}). The key scheduling function for the 80-bit key mode, KS_r^{80}, divides an 80-bit key $K_{(80)}$ into five 16-bit sub-keys $k_{i(16)}$ $(0 \le i < 5)$ and provides $wk_{i(16)} (0 \le i < 4)$ and $rk_{j(16)} (0 \le j < 2r)$ as follows:

Algorithm $KS_r^{80}(K_{(80)})$ **:**
$wk_0 \leftarrow k_0^L | k_1^R, wk_1 \leftarrow k_1^L | k_0^R, wk_2 \leftarrow k_4^L | k_3^R, wk_3 \leftarrow k_3^L | k_4^R$
for $i \leftarrow 0$ to $(r-1)$ do

$$(rk_{2i}, rk_{2i+1}) \leftarrow (con_{2i}^{80}, con_{2i+1}^{80}) \oplus \begin{cases} (k_2, k_3) \text{ (if } i \bmod 5 = 0 \text{ or } 2) \\ (k_0, k_1) \text{ (if } i \bmod 5 = 1 \text{ or } 4) \\ (k_4, k_4) \text{ (if } i \bmod 5 = 3), \end{cases}$$

where k_i^L and k_i^R are left and right half 8 bits of k_i, respectively, i.e., $k_{i(16)} = k_{i(8)}^L | k_{i(8)}^R$ and $k_{i(8)}^R$ contains the least significant bit of $k_{i(16)}$.

Key Schedule for 128-Bit Key Mode (KS_r^{128}). The key scheduling function for the 128-bit key mode, KS_r^{128}, divides a 128-bit key $K_{(128)}$ into eight 16-bit sub-keys $k_{i(16)}$ $(0 \le i < 8)$ and provides $wk_{i(16)} (0 \le i < 4)$ and $rk_{j(16)} (0 \le j < 2r)$ as follows:

Algorithm $KS_r^{128}(K_{(128)})$ **:**
$wk_0 \leftarrow k_0^L | k_1^R$, $wk_1 \leftarrow k_1^L | k_0^R$, $wk_2 \leftarrow k_4^L | k_7^R$, $wk_3 \leftarrow k_7^L | k_4^R$
for $i \leftarrow 0$ to $(2r-1)$ do
  if $(i+2) \bmod 8 = 0$ then
    $(k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7) \leftarrow (k_2, k_1, k_6, k_7, k_0, k_3, k_4, k_5)$
  $rk_i \leftarrow k_{(i+2) \bmod 8} \oplus con_i^{128}$

3 Design Rationale

In this section, we briefly describe design rationale of *Piccolo*.

Structure. *Piccolo* supports 64-bit block to fit standard applications, and 80 and 128-bit keys to achieve moderate security levels. The underlying structure is a variant of GFN that can easily support decryption function without much implementation cost and has light round functions.

Key Schedule. We adopt a permutation based key schedule which can significantly reduce the required number of gates. For instance, the registers for storing keys are not required and it leads the almost same gate requirement for each key size, in contrast to a key schedule requiring key state. While the drawback is security concern, by carefully choosing the permutation, it has enough immunity against attacks exploiting weakness of the key schedule such as related-key differential and MITM attacks. Note that, in our evaluation, key inputs are not required to be hard-wired, but are assumed to hold its values during the block operation.

Round Permutation. In order to improve diffusion property, *Piccolo* utilizes an 8-bit word based permutation between rounds instead of a 16-bit word based cyclic shift used in the standard GFN. Moreover, it demolishes the 16-bit word structure and thus improves the security against cryptanalysis exploiting strong word-based structure such as saturation attacks. We choose the specific one among several possibilities not to destroy the involution property in which the encryption process is identical to the decryption process when whitening and round keys are not introduced.

F-Function. The F-function consists of two S-box layers separated by a diffusion matrix without key additions before the second S-box layer. The S-box in the F-function has a 4-round iterative structure like GFN, and is extremely light. As shown in Fig. 4, each S-box consists of only four NOR gates, three XOR gates and one XNOR gate. Both the maximum differential probability (MDP) and the maximum linear probability (MLP) of the S-box are 2^{-2} which are optimal, and it has no fixed point. Moreover, it is suitable for efficient threshold implementation as discussed in Section 5. Furthermore, by using a standard PC, we obtain $2^{-9.3}$ and $2^{-8.0}$ as MDP and MLP of the F-function, respectively. While those figures are not optimal for a 16-bit bijective function, it is sufficient for our design, since *Piccolo* has enough differentially and linearly active F-functions over a certain number of rounds.

4 Security Analysis

In this section, we provide results on security analysis for *Piccolo*.

Differential Attack / Linear Attack [7,30]. We first show the minimum numbers of differentially and linearly active F-functions of G_r up to 30 rounds in Table 3. The figures in the table are obtained by an exhaustive search based on the algorithm given by [39]. Note that the minimum numbers for differentially and linearly active F-functions are the same due to the duality of differential and linear attacks and the similarity of G_r and G_r^{-1}. As explained in Section 3, MDP and MLP of the F-function are $2^{-9.3}$ and $2^{-8.0}$, respectively. Combining those results, *Piccolo* consisting of at least 7 or 8 rounds provide at least 7 or 8 active F-functions, and have no differential or linear trails whose probabilities are more than 2^{-64}, respectively. Thus, we expect that the full-round of *Piccolo* (25 and 31 rounds for *Piccolo*-80 and -128) has enough immunity against differential and linear attacks, since it has large security margin.

Boomerang-Type Attacks [42,23,6]. The boomerang-type attacks (including the boomerang, amplified boomerang and rectangle attacks) first divide the cipher into two sub-ciphers, then find a boomerang quartet with high probability. The probability of constructing a boomerang quartet is denoted as $\hat{p}^2\hat{q}^2$, where $\hat{p} = \sqrt{\sum_\beta \Pr^2[\alpha \to \beta]}$, and α and β are input and output differences for the first sub-cipher, and $\hat{q}$ for the second sub-cipher. $\hat{p}^2$ is bounded by the maximum differential trail probability, i.e., $\hat{p}^2 \leq \max_\beta \Pr[\alpha \to \beta]$, and $\hat{q}^2$ as well. Let p, q

Table 3. Min. # differentially and linearly active F-functions (single-key setting)

rounds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
min. # active F-functions	0	1	2	3	4	6	7	8	9	10	11	12	13	14	15
rounds	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
min. # active F-functions	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

be the maximum differential trail probability for the first and the second sub-ciphers. Then, p, q are bounded by multiplying the minimum number of active F-functions in each sub-cipher with MDP of the F-function. From Table 3, any combination of two sub-ciphers for *Piccolo* consisting of at least 9 rounds has at least 7 active F-functions in total. Hence, we conclude that the full-round of *Piccolo* is sufficiently secure against boomerang-type attacks.

Impossible Differential Attack [5]. An impossible differential attack is likely to be applied to a variant of GFN due to its slow diffusion. However, *Piccolo* utilizes the round permutation RP to achieve faster diffusion compared to a standard type-II GFN. Then, for both encryption and decryption sides, *Piccolo* requires only four rounds to be full diffusion, which is a property that all outputs are affected by all inputs. This implies that there exists at most 9-round impossible differential using a 16-bit truncated differential from the observation in [41]. We also search the longest impossible differential by modified $\mathcal{U}$-method [24] algorithm and found a 7-round impossible differential exploiting a 4-bit truncated differential. Therefore, we conclude that the full-round of *Piccolo* is expected to be secure against the impossible differential attack.

Related-Key Differential Attacks [9,8]. In the related-key setting, a distinguisher is allowed to use related-keys and usually uses key differentials to cancel out differentials in a data processing part. While the practical impact of related-key differential attacks is still controversial, we care about it from a pessimistic (designers') point of view. To evaluate the resistance to it, we follow an approach presented in [10]. In other words, we evaluate the immunity against related-key differential attacks by counting the minimum number of differentially active F-functions in the related-key setting. Table 4 shows the minimum numbers of differentially active F-functions for the 80 and the 128-bit key modes up to 20 rounds. Unlike the attacks under the single-key setting, the total number of active F-functions for the related-key differential attacks may vary according to the starting round. However, in our evaluations, those differences are at most 2 active F-functions, even if the starting round is changed. Consequently, we obtain that over 14 and 16 rounds for *Piccolo*-80 and -128 have at least 7 differentially active F-functions in the related-key setting, respectively.

Moreover, we consider related-key boomerang/rectangle attacks [8]. Similarly to non related-key boomerang-type attacks, we evaluate the security in the worst case that an attacker can use pq instead of $\hat{p}^2\hat{q}^2$ for the probability of a boomerang quartet. As a result, we confirmed that over 17 and 21 rounds of *Piccolo*-80 and -128 provide enough (seven) differentially active F-functions in this setting.

Table 4. Min. # differentially active F-functions (related-key setting)

starting round i \ rounds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
for *Piccolo*-80 encryption																				
$i \bmod 5 = 0$	0	0	0	0	0	2	3	4	4	5	5	6	7	7	7	8	9	10	11	11
$i \bmod 5 = 1$	0	0	0	0	1	2	3	4	4	5	6	6	7	7	8	9	9	10	11	11
$i \bmod 5 = 2$	0	0	0	0	1	2	3	3	4	6	6	6	7	7	9	9	9	10	10	12
$i \bmod 5 = 3$	0	0	0	0	1	2	2	3	4	5	5	6	6	7	8	8	9	9	10	11
$i \bmod 5 = 4$	0	0	0	0	0	0	2	3	4	5	6	6	7	7	7	7	9	10	11	11
for *Piccolo*-80 decryption																				
$i \bmod 5 = 0$	0	0	0	0	1	2	2	3	4	5	5	6	6	7	8	8	9	9	10	11
$i \bmod 5 = 1$	0	0	0	0	1	2	3	3	4	6	6	6	7	7	9	9	9	10	10	12
$i \bmod 5 = 2$	0	0	0	0	1	2	3	4	4	5	6	6	7	7	8	9	9	10	11	11
$i \bmod 5 = 3$	0	0	0	0	0	2	3	4	4	5	5	6	7	7	7	8	9	10	11	11
$i \bmod 5 = 4$	0	0	0	0	0	0	2	3	4	5	6	6	7	7	7	7	9	10	11	11
for *Piccolo*-128 encryption																				
$i \bmod 4 = 0$	0	0	0	0	0	0	0	1	3	3	4	5	5	6	7	7	8	9	10	10
$i \bmod 4 = 1$	0	0	0	0	0	0	1	2	3	3	4	5	5	6	7	7	8	9	10	11
$i \bmod 4 = 2$	0	0	0	0	0	0	1	2	2	3	4	4	5	6	6	7	7	9	9	9
$i \bmod 4 = 3$	0	0	0	0	0	1	1	1	2	3	4	5	5	6	7	7	8	9	9	10
for *Piccolo*-128 decryption																				
$i \bmod 4 = 0$	0	0	0	0	0	1	1	2	3	3	4	5	5	6	6	7	8	9	9	11
$i \bmod 4 = 1$	0	0	0	0	0	0	1	2	3	3	4	4	5	6	7	7	8	9	9	9
$i \bmod 4 = 2$	0	0	0	0	0	0	0	1	2	3	4	5	5	6	7	7	7	9	10	10
$i \bmod 4 = 3$	0	0	0	0	0	0	1	1	2	3	4	5	5	6	7	7	8	9	10	10

Furthermore, we take related-key impossible differential attacks [22] into account. Consequently, by using modified $\mathcal{U}$-method, we found an 11 and a 17-round impossible differential distinguisher using an 8-bit truncated differential for *Piccolo*-80 and -128 in the related-key setting, respectively, and they are the longest in our evaluation. Therefore, we conclude that the full-round *Piccolo* is expected to be resistant to those attacks.

Meet-in-the-Middle Attack [12]. Three-subset meet-in-the-middle (MITM) cryptanalysis [12] is a recent attack on blockciphers. This attack works well for blockciphers having a simple key schedule and slow diffusion. Indeed, KTAN-TAN and GOST have been theoretically broken by this attack [12,21]. Since *Piccolo* consists of the permutation based key scheduling and a variant of GFN, evaluating the resistance against this attack is important.

Similarly to data difference, *Piccolo* requires 4 rounds to non-linearly diffuse any round-key difference to all output data in the data processing part, i.e., any round-key bits of the i-th round non-linearly affect all input of the $(i-3)$-th round and all output of the $(i+3)$-th round. Thus, we assume that an attacker might construct an 8-round *indirect-partial matching* [3] and a 4-round *initial structure* [37] in the worst case. Besides, we even allow the attacker to use *code book* and *splice and cut* techniques [2]. In this worst setting, *Piccolo*-80 and -128 without whitening keys have neutral words up to 19 and 23 consecutive rounds, respectively. We expect that the attacked rounds obtained by this observation are upper bounds on the security against the three-subset MITM attack, since the given assumptions are sufficiently strong. Moreover, we attempt to construct

actual attacks to obtain the lower bounds on the security. As a result, the *Piccolo*-80 and -128 without whitening keys reduced to 14 and 21 rounds can be attacked by the three-subset MITM attacks, respectively. Since *Piccolo* actually has whitening keys, it is obviously stronger than the variants evaluated above. Thus, we conclude that *Piccolo* has enough immunity against the three-subset MITM attack.

Other Attacks. We also consider other attacks including a slide, a saturation, an interpolation, a higher order differential, a truncated differential, and an algebraic attack. Though the details of the evaluations for those attacks are omitted due to the page limitation, consequently, we expect that none of them work better than the previously explained attacks.

5 Implementation Aspects

This section provides results on compact hardware implementation of *Piccolo* with novel implementation techniques, showing two types of implementations: a round-based implementation and a serialized implementation. While one round function is processed within one clock cycle in a round-based implementation, only a fraction of one round is treated in a clock cycle in a serialized implementation to realize the low-power and low-area implementation.

5.1 Optimization in Key Scheduling Part

The key scheduling part of *Piccolo* can be implemented by using multiplexers without flip-flops which have high area requirement, in a way similar to the implementation of GOST and KTANTAN [35,13]. Actually, our round-based implementation of *Piccolo*-80 needs only 32-bit wide 3-to-1 MUX to select the appropriate round key. For a serialized implementation, we require a 4-bit wide 20-to-1 MUX to select the right chunk of the round key.

In our evaluation, key inputs are assumed to hold those values during the block process, but are not required to be hard-wired. Therefore, our results do not contain registers for storing keys. If such registers are needed, around 360 and 576 extra GE are required for Piccolo-80 and -128, respectively. Moreover, if we use hard-wired key, we can reduce around 85 and 114 GE from the round-based implementations, also about 67 and 104 GE from the serialized implementations for *Piccolo*-80 and -128, respectively.

5.2 Optimization in Data Processing Part

A round-based implementation of *Piccolo* can be done straightforwardly. Note that we use scan flip-flops for the data state, which take both an input and an output of a round function as inputs.

On the other hand, a serialized implementation has many variety. Our serialized implementation is based on 4-bit shift registers in the similar way as [18]. The 4-bit data path for *Piccolo*-80 is described in Fig. 5.

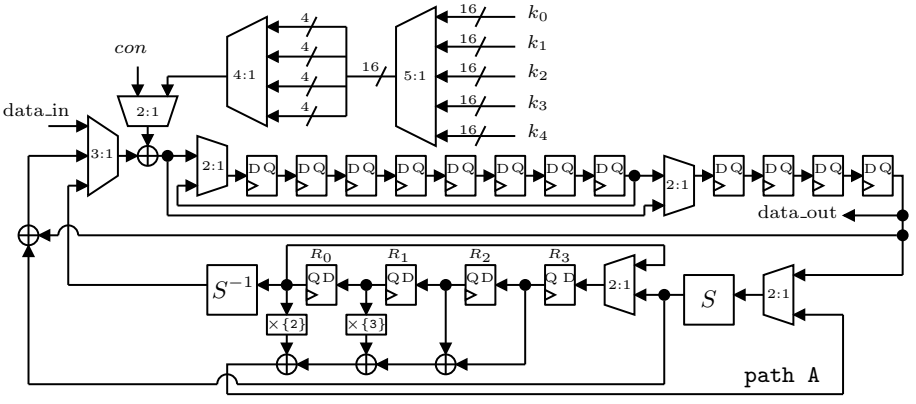

Fig. 5. Data path of our serialized implementation

In our serialized implementation, firstly outputs of the first S-box are set to the registers (R_0, R_1, R_2, R_3) described in Fig. 5. In the next four clock cycles, each row of the diffusion matrix is updated in order by rotating the registers (R_0, R_1, R_2, R_3). Simultaneously, the outputs of the matrix are input to S-box S through `path A`, then the outputs of the F-function are obtained. In the next four clock cycles, the inputs of the F-function are recovered in order through S^{-1} which is the inversion of S. At the same time, the outputs of the first S-box layer of the next F-function are set to the registers (R_0, R_1, R_2, R_3). Therefore, this implementation requires 8 clock cycles per F-function, and thus 16 clock cycles per round. We emphasize that our serialized implementation does not require additional registers for storing intermediate values of the F-functions by appending S^{-1} which costs only 12 GE.

5.3 Hardware Performance

Table 5 shows the detailed implementation figures of the round-based and the serialized implementations of *Piccolo*-80 and -128.

We designed hardware implementations of *Piccolo* in Verilog-HDL and synthesized the designs to a 0.13 μm standard cell library. We used *VCS version 2006.06* for simulation and *Design Compiler version 2007.03-SP3* for synthesis. One GE is equivalent to the area of a 2-way NAND.

In a recent trend, the implementation of lightweight blockciphers uses a scan flip-flop instead of a combination of a D flip-flop and a 2-to-1 MUX [13,35,36] to reduce the gate requirement. In our evaluation environment, a D flip-flop and a 2-to-1 MUX cost 4.5 and 2.0 GE, respectively, while a scan flip-flop costs 6.25 GE. Thus, we can save 0.25 GE per bit of storage by using this implementation technique. Moreover, the library we used has the 4-input AND-NOR and 4-input OR-NAND gates with two inputs inverted as described in Fig. 6. The outputs of these cells are corresponding to those of XOR or XNOR when the inputs X, Y are set as shown in Fig. 6. Thus, we can use these cells instead of XOR or

Table 5. Implementation figures for Piccolo

	Piccolo-80		*Piccolo*-128	
	serial	round	serial	round
cycles per block	432	27	528	33
throughput @ 100 kHz (kbps.)	14.81	237.04	12.12	193.94
Area [GE] sum	683.00	1,135.25	757.75	1,196.50
Key scheduling	95	72	135	120
Data state	309	344	309	344
S-box/S-box^{-1}	24	192	24	192
Matrix	34	208	34	208
Key XOR	8*	64	8*	64
Constants XOR	-*	40	-*	40
F-func. output XOR	8	64	8	64
MUX	24	72	24	72
Others/Control	181.00	79.25	215.75	92.50

∗: XOR for round keys and constants is shared

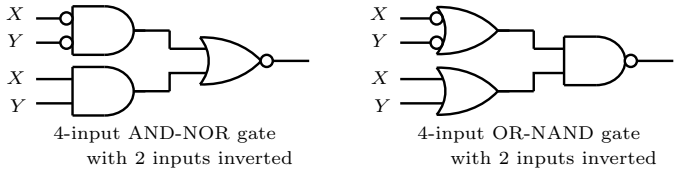

Fig. 6. 4-input AND-NOR and 4-input OR-NAND gates with 2 inputs inverted, which correspond to XOR and XNOR gate

XNOR cells. Since both cells cost 2 GE instead of 2.25 GE required for XOR or XNOR, we can save 0.25 GE per an XOR or XNOR gate. We employed the above mentioned implementation techniques in our evaluation.

5.4 Security against Side Channel Attacks

A provably secure countermeasure against first order side-channel attacks called threshold implementations [33,34] can be applied to *Piccolo*. In threshold implementations, at least three shares are necessary for any nonlinear function. The S-box of *Piccolo* defined in Section 2 is chosen to belong to the alternating group A_{16}, where a 4×4 bijection can be decomposed using quadratic bijections [14]. Therefore, for the S-box of *Piccolo*, the masking method can be applied using only three shares, which leads efficient threshold implementations of *Piccolo*.

6 Conclusion

In this paper, we have presented a lightweight blockcipher consisting of a variant of generalized Feistel network with a permutation based key schedule. Despite several desirable implementation properties for a combination of Feistel-type

structure with a permutation based key schedule, the ciphers having such structures are likely to be vulnerable to attacks. The proposed cipher *Piccolo* employs several new design approaches including the half-word based round permutation and the effective permutation for key expanding to avoid known attacks without loosing efficiency on both power and energy consumptions. Consequently, *Piccolo* achieves not only notably compact implementation but also high security.

Acknowledgments. The authors would like to thank the anonymous reviewers for their helpful comments.

References

1. Akishita, T., Hiwatari, H.: Very compact hardware implementations of the blockcipher CLEFIA. Sony corporation (June 2011), `http://www.sony.co.jp/Products/cryptography/clefia/download/data/clefia-hw-compact-20110615.pdf`
2. Aoki, K., Sasaki, Y.: Preimage attacks on one-block MD4, 63-step MD5 and more. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 103–119. Springer, Heidelberg (2009)
3. Aoki, K., Sasaki, Y.: Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 70–89. Springer, Heidelberg (2009)
4. Badel, S., Dagtekin, N., Nakahara, J., Ouafi, K., Reffé, N., Sepehrdad, P., Susil, P., Vaudenay, S.: ARMADILLO: A multi-purpose cryptographic primitive dedicated to hardware. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 398–412. Springer, Heidelberg (2010)
5. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 12–23. Springer, Heidelberg (1999)
6. Biham, E., Dunkelman, O., Keller, N.: The rectangle attack - rectangling the Serpent. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 340–357. Springer, Heidelberg (2001)
7. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer, Heidelberg (1993)
8. Biham, E., Dunkelman, O., Keller, N.: Related-key boomerang and rectangle attacks. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 507–525. Springer, Heidelberg (2005)
9. Biham, E., Dunkelman, O., Keller, N.: A unified approach to related-key attacks. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 73–96. Springer, Heidelberg (2008)
10. Biryukov, A., Nikolić, I.: Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 322–344. Springer, Heidelberg (2010)
11. Bogdanov, A., Knudsen, L., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
12. Bogdanov, A., Rechberger, C.: A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 229–240. Springer, Heidelberg (2011)

13. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — A family of small and efficient hardware-oriented block ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
14. De Cannière, C., Nikov, V., Nikova, S., Rijmen, V.: S-box decompositions for SCA-resisting implementations. In: Poster Session of CHES 2010 (2010)
15. De Cannière, C., Preneel, B.: TRIVIUM. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 244–266. Springer, Heidelberg (2008)
16. FIPS, Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197
17. FIPS, Data Encryption Standard. Federal Information Processing Standards Publication 46
18. Hämäläinen, P., Alho, T., Hännikäinen, M., Hämäläinen, T.D.: Design and implementation of low-area and low-power AES encryption hardware core. In: DSD, pp. 577–583. IEEE Computer Society, Los Alamitos (2006)
19. Hell, M., Johansson, T., Maximov, A., Meier, W.: The Grain Family of Stream Ciphers. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 179–190. Springer, Heidelberg (2008)
20. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S.: HIGHT: A new block cipher suitable for low-resource device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)
21. Isobe, T.: A single-key attack on the full GOST block cipher. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 290–305. Springer, Heidelberg (2011)
22. Jakimoski, G., Desmedt, Y.: Related-key differential cryptanalysis of 192-bit key AES variants. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 208–221. Springer, Heidelberg (2004)
23. Kelsey, J., Kohno, T., Schneier, B.: Amplified boomerang attacks against reduced-round MARS and Serpent. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 75–93. Springer, Heidelberg (2001)
24. Kim, J., Hong, S., Sung, J., Lee, C., Lee, S.: Impossible differential cryptanalysis for block cipher structures. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT 2003. LNCS, vol. 2904, pp. 82–96. Springer, Heidelberg (2003)
25. Knudsen, L., Leander, G., Poschmann, A., Robshaw, M.J.B.: PRINTcipher: A block cipher for IC-printing. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 16–32. Springer, Heidelberg (2010)
26. Koo, B., Hong, D., Kwon, D.: Related-key attack on the full HIGHT. In: Pre-Proceedings of ICISC 2010. Springer, Heidelberg (2010)
27. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New lightweight DES variants. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 196–210. Springer, Heidelberg (2007)
28. Lim, C.H., Korkishko, T.: mCrypton – A lightweight block cipher for security of low-cost RFID tags and sensors. In: Song, J.-S., Kwon, T., Yung, M. (eds.) WISA 2005. LNCS, vol. 3786, pp. 243–258. Springer, Heidelberg (2006)
29. Lim, C.H.: A Revised Version of CRYPTON - CRYPTON V1.0 -. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 31–45. Springer, Heidelberg (1999)
30. Matsui, M.: Linear cryptanalysis of Data Encryption Standard. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
31. Moradi, A., Poschmann, A., Ling, S., Paar, C., Wang, H.: Pushing the limits: A very compact and a threshold implementation of AES. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 69–88. Springer, Heidelberg (2011)

32. National Soviet Bureau of Standards, Information Processing System - Cryptographic Protection - Cryptographic Algorithm GOST 28147-89
33. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 529–545. Springer, Heidelberg (2006)
34. Nikova, S., Rijmen, V., Schläffer, M.: Secure hardware implementation of nonlinear functions in the presence of glitches. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 218–234. Springer, Heidelberg (2009)
35. Poschmann, A., Ling, S., Wang, H.: 256 bit standardized crypto for 650 GE – GOST revisited. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 219–233. Springer, Heidelberg (2010)
36. Rolfes, C., Poschmann, A., Leander, G., Paar, C.: Ultra-lightweight implementations for smart devices – security for 1000 gate equivalents. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 89–103. Springer, Heidelberg (2008)
37. Sasaki, Y., Aoki, K.: Finding preimages in full MD5 faster than exhaustive search. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009)
38. Satoh, A., Morioka, S.: Hardware-Focused Performance Comparison for the Standard Block Ciphers AES, Camellia, and Triple-DES. In: Boyd, C., Mao, W. (eds.) ISC 2003. LNCS, vol. 2851, pp. 252–266. Springer, Heidelberg (2003)
39. Shirai, T., Araki, K.: On generalized Feistel structures using the diffusion switching mechanism. IEICE Trans. Fundamentals E91-A(8), 2120–2129 (2008)
40. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-Bit Blockcipher CLEFIA (Extended Abstract). In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer, Heidelberg (2007)
41. Suzaki, T., Minematsu, K.: Improving the generalized Feistel. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 19–39. Springer, Heidelberg (2010)
42. Wagner, D.: The boomerang attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)

A Test Vectors

We give test vectors of *Piccolo* for each key length. The data are represented in hexadecimal form.

80-bit key:

key	`00112233 44556677 8899`
plaintext	`01234567 89abcdef`
ciphertext	`8d2bff99 35f84056`

128-bit key:

key	`00112233 44556677 8899aabb ccddeeff`
plaintext	`01234567 89abcdef`
ciphertext	`5ec42cea 657b89ff`