KLEIN: A New Family of Lightweight Block Ciphers

Zheng Gong[1], Svetla Nikova[2,3], and Yee Wei Law[4]

[1] School of Computer Science, South China Normal University, China
cis.gong@gmail.com
[2] Faculty of EWI, University of Twente, The Netherlands
[3] Dept. ESAT/SCD-COSIC, Katholieke Universiteit Leuven, Belgium
s.i.nikova@utwente.nl
[4] Department of EEE, The University of Melbourne, Australia
yee.wei.law@gmail.com

Abstract. Resource-efficient cryptographic primitives are essential for realizing both security and efficiency in embedded systems like RFID tags and sensor nodes. Among those primitives, lightweight block cipher plays a major role as a building block for security protocols. In this paper, we describe a new family of lightweight block ciphers named KLEIN, which is designed for resource-constrained devices such as wireless sensors and RFID tags. Compared to related proposals, KLEIN has advantage in the software performance on legacy sensor platforms, while its hardware implementation can be compact as well.

1 Introduction

With the rapid advances in wireless communication and embedded systems, we are becoming increasingly dependent on the so-called pervasive computing, evidence of which can be found in ubiquitous smart cards, RFID tags, public transport systems, smart meters, etc. Wireless sensor networks (WSNs), due to its potential in pushing the envelope of pervasive computing to areas such as environment monitoring, military surveillance and healthcare, is attracting more and more attention. When choosing security algorithms for resource-limited devices the implementation costs should be taken into account. Symmetric-key algorithms, especially block ciphers, still play an important role in the security of embedded systems. Moreover recent results have shown that lightweight block ciphers can be used not only for encryption, but also for hash [4] and authentication [21] on devices with highly constrained resources. For security and performance concerns, some types of sensors are equipped with a hardware implementation of AES-128 [14], e.g. the Chipcon CC2420 transceiver chip [7]. But for resource-constrained devices, AES could be too expensive, despite the various approaches that have been proposed to reduce the costs of AES hardware and software implementations [20,24,25,31].

In the literature, quite a few lightweight block ciphers with various design strategies have been proposed [3,6,13,18,22,27,33,35,42,52]. Skipjack is a lightweight block cipher designed by the U.S. National Security Agency (NSA) for embedded applications [42]. The algorithm of Skipjack has an 80-bit key with a 64-bit block length based on an unbalanced Feistel network. NOEKEON is a hardware-efficient block cipher, which

A. Juels and C. Paar (Eds.): RFIDSec 2011, LNCS 7055, pp. 1–18, 2012.

was proposed by Daemen *et al.* [13] and submitted to the NESSIE project in 2000. HIGHT was designed by Hong *et al.* [22] as a generalized Feistel-like cipher, which is suitable for low-resource devices. mCrypton [35] is designed by following the overall architecture of Crypton [34] but with redesign and simplifications of each component function to enable much compact implementation in both hardware and software. SEA is a software-oriented block cipher which was proposed by Standaert *e*t al. [52]. At FSE 2007, Leander *et al.* [33] proposed a family of new lightweight variants of DES, which are called DESL and DESXL. The main idea of the new variants of DES is to use just one S-box recursively, instead of eight different S-boxes. Bogdanov *et al.* proposed [3] an ultra-lightweight block cipher which is called PRESENT. The design of PRESENT is extremely hardware efficient, since it uses a fully wired diffusion layer without any algebraic unit. KATAN and KTANTAN are designed as a family of ultra-lightweight block ciphers by De Cannière *et al.* [6]. Both KATAN and KTANTAN use an 80-bit key length with 32, 48, or 64-bit block size, while KTANTAN is more compact in hardware since its key will be unchangeably burnt on devices. In [18], Engels *et al.* proposed a novel ultra-lightweight cryptographic algorithm with 256-bit key length and 16-bit block size, referred to as Hummingbird, for resource-constrained devices.

The security of any block cipher should be extensively analyzed before its wide implementation. Biham *et al.* have discovered an impossible differential attack on 31 of the 32 rounds [1] of Skipjack. A truncated differential attack was also published against 28 rounds of Skipjack by Knudsen *et al.* [29]. Granboulan [23] presented a revised result in the differential analysis of Skipjack. By exploiting its periodic key schedule, a complementation slide attack is mounted on the full 32 rounds of Skipjack [46]. The attack requires only $2^{32.5}$ known texts and 2^{44} encryptions of Skipjack. In a NESSIE report, Knudsen and Raddum [28] showed that "indirect mode" NOEKEON was still vulnerable to certain peculiar kinds of related-key cryptanalysis, and discovered weaknesses in NOEKEON-variant ciphers which cast doubt on the design strategy behind NOEKEON and thus on its security. As a result NOEKEON was not selected by NESSIE. Although PRESENT has a hardware-efficient diffusion layer, different attacks have been applied to the reduced-round variants of PRESENT due to its diffusion property, e.g. the weak key attack [43,44], the linear attack [8] and the saturation attack [9]. Recently, Bogdanov and Rechberger [5] have proposed a meet-in-the-middle attack on KTANTAN. At FSE 2011, Saarinen [50] presented a chosen-IV, chosen-message differential attack which can break full-round Hummingbird in practice.

The performance of a block cipher is also a key factor for resource-constrained devices. Efficiency in hardware is a major design criterion for lightweight block ciphers. The area in *gate equivalents* (GE) is often used as a measure for the compactness of the hardware implementation. Generally speaking, one GE is equal to the area which is required by two-input NAND gate with the lowest driving strength of the appropriate technology [45]. PRESENT, for example, has a compact implementation with 1570 GE in a 64-bit datapath [4], as well as a very lightweight implementation with 1000 GE [49]. mCrypton and DESXL are also competitive since they are close to the 2000 GE barrier. HIGHT is less attractive since its area in GE is over 3000 GE, which is less competitive to the best AES implementations by Moradi *et al.* [40] (with 2400 GE), Hamalainen *et al.* [24] (with 3100 GE) and Feldhofer *et al.* [20] (with 3400 GE). PRINTCipher

[27], GOST [48], KATAN and KTANTAN [6] are among the most hardware-efficient, requiring less than 1000 GE.

Usually, sensors have better power and hardware capabilities than RFID tags. Since software implementations incur zero hardware manufacturing cost and are flexible to maintain, it is believed that software-efficient block ciphers are more practical for sensors. In this paper, a new family of block ciphers called *KLEIN* is designed for resource-constrained devices. Compared to related proposals, KLEIN has the advantage of the software performance on legacy sensor platforms and at the same time its hardware implementation can also be compact. Our security analysis shows that KLEIN has a conservative security margin against various cryptanalyses.

The remainder of this paper is organized as follows. Section 2 describes the design rationale and the specification of the KLEIN family. In Section 3, the security of KLEIN is analyzed by considering known attacks. Compared to related lightweight proposals, a detailed performance of KLEIN is discussed in Section 4. Section 5 concludes the paper.

2 Specification of KLEIN

In this section we specify the cipher structure of KLEIN. Also the design principles will be discussed, which are followed during the design process of KLEIN. For each of the components of KLEIN, our choices will be motivated to achieve the well balanced trade-off between performance and security. The test vectors of KLEIN can be found in Appendix A.

2.1 Structure of KLEIN

The structure of KLEIN is a typical Substitution-Permutation Network (SPN), which is also used in many advanced block ciphers, e.g. AES and PRESENT. In our first estimation for obtaining a reasonable security margin and asymmetric iteration, we choose the number of rounds N_R as 12/16/20 for KLEIN-64/80/96 respectively. A high-level description of the KLEIN encryption routine is described in Figure 1.

$sk^1 \leftarrow$ KEY;
STATE $\leftarrow$ PLAINTEXT;
for $i = 1$ *to* N_R do
  $AddRoundKey$(STATE, sk^i);
  $SubNibbles$(STATE);
  $RotateNibbles$(STATE);
  $MixNibbles$(STATE);
  $sk^{i+1} = KeySchedule(sk^i, i)$;
end for
CIPHERTEXT $\leftarrow AddRoundKey$(STATE, sk^{N_R+1});

Fig. 1. The encryption routine of KLEIN

Note that many lightweight block ciphers are proposed to use only the counter mode and hence, the implementation costs of decryptions can be avoided. In the design of KLEIN, its lightweight property should also take the decryption algorithm into consideration without fixing on any cipher mode.

2.2 The Round Transformation

The input and output of KLEIN are considered to be one-dimensional byte arrays. During the round transformation, all the operations can be optimized with byte-oriented algorithms.

The SubNibbles Step. Since the $AddRoundKey(x, y)$ step in the round transformation is simply $x \oplus y$, the input text will be XORed with the i-th round key sk^i (where $i \in [1, N_R]$) before the SubNibbles step. In the SubNibbles step, the XORed results will be divided into 16 of 4-bit nibbles and input to the same 16 S-boxes. The KLEIN S-box S is a 4×4 involutive permutation. The non-linear permutation executed by S is described in Table 1. The implementation costs of such a 4-bit S-box is much lower than that of an 8-bit S-box either by hardware or by software. By choosing an involutive S-box, we can also save the implementation costs for its inverse. Since the same S-boxes are used in the SubNibbles step, it allows a serialization of the design for an extremely small footprint. Moreover, we just need to provide one single side-channel protection for the S-box. Thus the overhead of an extra protection on its inverse is unnecessary.

Table 1. The 4-Bit S-box used in KLEIN

Input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Output	7	4	A	9	1	F	B	0	C	3	2	6	8	E	D	5

Since the SubNibbles step is the only non-linear layer in KLEIN, a natural requirement is an optimal resistance against linear and differential cryptanalyses. Therefore the choice of the S-box S fulfills the following conditions.

1. The S-box satisfies $\mathsf{S}(\mathsf{S}(x)) = x, x \in \mathbb{F}_2^4$, thus it can be used both in the encryption and in the decryption.
2. The S-box has no fixed points, i.e. $\mathsf{S}(x) \neq x, x \in \mathbb{F}_2^4$.
3. For any non-zero input difference $\Delta_I \in \mathbb{F}_2^4$ and output difference $\Delta_O \in \mathbb{F}_2^4$, it holds that
$$\sharp\{x \in \mathbb{F}_2^4 | \mathsf{S}(x) + \mathsf{S}(x + \Delta_I) = \Delta_O\} \leq 4. \quad (1)$$
Furthermore, if $wt(\Delta_I) = wt(\Delta_O) = 1$, we have
$$\sharp\{x \in \mathbb{F}_2^4 | \mathsf{S}(x) + \mathsf{S}(x + \Delta_I) = \Delta_O\} \leq 2. \quad (2)$$
4. For any non-zero $a, b \in \mathbb{F}_2^4$, it holds that
$$|S_b^{\mathcal{W}}(a)| = |\sum_{x \in \mathbb{F}_2^4} (-1)^{b \cdot \mathsf{S}(x) + a \cdot x}| \leq 8. \quad (3)$$

Furthermore, if $wt(a) = wt(b) = 1$, we have

$$|S_b^{\mathcal{W}}(a)| = |\sum_{x \in \mathbb{F}_2^4} (-1)^{b \cdot \mathsf{S}(x) + a \cdot x}| \leq 4. \quad (4)$$

The 4-bit S-box used in PRESENT satisfies $\sharp\{x \in \mathbb{F}_2^4 | \mathsf{S}(x) + \mathsf{S}(x + \Delta_I) = \Delta_O\} = 0$ if $wt(\Delta_I) = wt(\Delta_O) = 1$, which assures a better avalanche effect [3]. However, the PRESENT S-box is not an involution. According to our exhaustive search result, there is no such an involutive 4-bit S-box that can satisfy this additional property.

For each input differential Δ_I, the maximum probability of any output differential Δ_O is up to $4/16 = 2^{-2}$. Let p be the probability of a linear characteristic. The correlation of the linear characteristic over S is given by $q = (2p - 1)^2$ [38]. From the input-output correlation of S, it is straightforward that any linear characteristic over S has a correlation of at most $(2 \times \frac{4}{16} - 1)^2 = 2^{-2}$.

The RotateNibbles Step. After the SubNibbles step, 16 nibbles $b_0^i, b_1^i, \cdots, b_{15}^i$ will be rotated left two bytes during the i-th round where $i \in [1, N_R]$. The RotateNibbles step is illustrated in Figure 2. The inverse operation will be simply rotate right two bytes per round. Nevertheless, the RotateNibbles step can also be combined with the MixNibbles step to avoid the hardware or software costs.

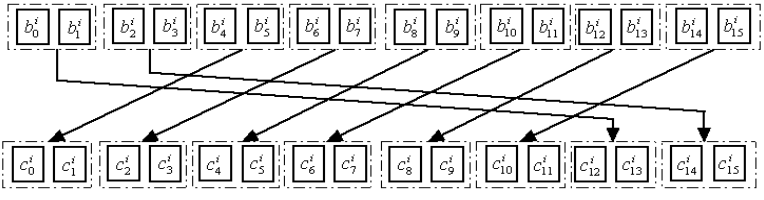

Fig. 2. The RotateNibbles step

The MixNibbles Step. The MixNibbles step is a bricklayer permutation of the state. The i-th round input nibbles $\{c_0^i, c_1^i, \cdots, c_{15}^i\}$ will be divided into 2 tuples, which will be proceeded the same as the MixColumns step in Rijndael. The tuples of the state are considered as polynomials over $\mathbb{F}_2^8$ and multiplied modulo $x^4 + 1$ with a fixed polynomial $c(x) = 03 \cdot x^3 + 01 \cdot x^2 + 01 \cdot x + 02$. The inverse is also a fixed multiplication polynomial $d(x) = 0\mathrm{B} \cdot x^3 + 0\mathrm{D} \cdot x^2 + 09 \cdot x + 0\mathrm{E}$. The output of the MixNibbles step will be the intermediate state s^{i+1} for the next round transformation.

Note that the balance between the diffusion property and the software performance let us make this choice. Although bit-shifting operations are often used in the diffusion layer of many lightweight block ciphers (e.g. PRESENT and NOEKEON), efficiency is lost in software implementations. From the number of active Sboxes, it seems a better choice that the MixNibbles step chooses a matrix multiplication in $GF(2^4)$. However, a byte-oriented matrix multiplication has advantages in the software implementations for 8-bit processors (e.g. Skipjack). By using the similar implementation of the

MixColumns step for 8-bit processors [14], we can use just one 256-byte look-up table to optimize the MixNibbles step. Also the same look-up table can be used to optimize its inverse. After the 12/16/20 rounds of KLEIN-64/80/96, the MixNibbles step can still provide a high number of active Sboxes for the security of KLEIN. The property of the MixColumns step of Rijndael has been well analyzed, the details can be found in the literature [12,14,16].

Key schedule. For round transformations, all practical block ciphers use varied key schedules to expand a relative small master key to a series of dependent round keys. Since KLEIN will be used to construct block-cipher-based hash functions and message authentication codes, the key schedule should be agile even if keys are frequently changed. On the other hand, the key schedule should also consider a proper complexity for the security. To avoid the potential related-key weakness whilst balancing the performance, the key schedule of KLEIN is designed as follows.

1. Input: a 64/80/96-bit master key mk for KLEIN-64/80/96.
2. Key scheduling: Let i be the round counter of KLEIN-64/80/96. In the first round so that $i = 1$, the initial subkey $sk^1 = mk = sk_0^1||sk_1^1||\cdots||sk_t^1$ where $t = 7/9/11$ for KLEIN-64/80/96. For KLEIN-64, the $(i+1)$-th subkey sk^{i+1} can be derived from the i-th subkey sk^i as follows.
   (a) Divide the i-th subkey sk^i into two byte-oriented tuples (a, b), such that $a = (sk_0^i, sk_1^i, \cdots, sk_{\lfloor \frac{t}{2} \rfloor}^i)$ and $b = (sk_{\lceil \frac{t}{2} \rceil}^i, sk_{\lceil \frac{t}{2} \rceil + 1}^i, \cdots, sk_t^i)$. For KLEIN-64, we have $a = (sk_0^i, sk_1^i, sk_2^i, sk_3^i)$ and $b = (sk_4^i, sk_5^i, sk_6^i, sk_7^i)$.
   (b) Cycling left shift one byte position in (a, b), obtain $a' = (sk_1^i, \cdots, sk_{\lfloor \frac{t}{2} \rfloor}^i, sk_0^i)$ and $b' = (sk_{\lceil \frac{t}{2} \rceil + 1}^i, \cdots, sk_t^i, sk_{\lceil \frac{t}{2} \rceil}^i)$ for the next step. For KLEIN-64, we have $a' = (sk_1^i, sk_2^i, sk_3^i, sk_0^i)$ and $b' = (sk_5^i, sk_6^i, sk_7^i, sk_4^i)$.
   (c) Swap the tuple (a', b') with a Feistel-like structure, such that $a'' = b'$ becomes the left tuple, whilst $b'' = a' \oplus b'$ becomes the right tuple.
   (d) XOR round counter i with the third byte in the left tuple a'', and substitute the second and the third bytes of the right tuple b'' by using the KLEIN S-box S.
3. Output: iteratively execute the above step for different key lengths, truncate the leftmost 64 bits of subkey sk^i for the i-th round transformation.

Figure 3 illustrates the KeySchedule algorithm of KLEIN-64. The key schedule of KLEIN is feasible for different key sizes. To save the memory for storing intermediate values, the subkeys of KLEIN can be generated during each round transformation. During the performance tuning on sensors, we observed that the on-the-fly key schedule of KLEIN is more resource-efficient than the traditional optimization such that all subkeys are computed in advance. Also the Feistel-like structure provides more complexities to resist weak key attacks, which was found on the PRESENT block cipher recently [8,43]. For simplicity, we only use an incremental round counter as the additive constant to avoid the slide attack. Like some other block cipher schemes, those round counters in KLEIN can also be defined by a recursion rule or an LFSR sequence in $GF(2^8)$ to avoid the potential complementation properties.

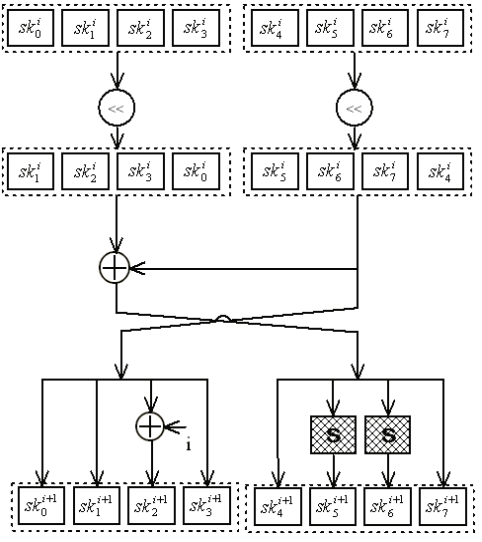

Fig. 3. The KeySchedule algorithm of 64-bit key length

3 Security Analysis

In this section we will present a security analysis of KLEIN, showing its resistance against various cryptanalytic attacks.

3.1 Linear and Differential Attacks

The resistance of linear and differential attacks of a block cipher is mainly based on the branch number, i.e. the number of active S-boxes in a certain number of rounds. In Rijndael, the authors use the Maximum Distance Separable (MDS) code to achieve the maximal branch number in a small number of rounds. By combining the RotateNibbles and MixNibbles steps, KLEIN can achieve a balance between the minimum number of active S-boxes and the software performance for resource-constrained devices.

Theorem 1. *Any four-round differential characteristic of KLEIN has a minimum of 15 active S-boxes.*

Proof. The MixColumns step in Rijndael is based on a Maximum Distance Separable code and the distance between any two distinct words called branch number is 5 [14]. Since we use the same matrix multiplication in the MixNibbles step of KLEIN, the branch number of MixNibbles is also 5. Since MixNibbles is computed with multiplications in $GF(2^8)$, an active byte in the diffusion layer of KLEIN implies one or two nibbles (i.e. the leftmost and the rightmost 4 bits) are active. For simplicity, we assume every active byte only has one active nibbles. Let $\Delta_i = L_i||R_i$ be the i-th round input difference characteristic, where L_i and R_i denote the left and the right 4-byte tuples respectively. The differential patterns of four-round KLEIN can be analyzed as follows.

- If there is 1 non-zero byte in L_1, it will be at least one active S-box in the first round. After the Rotate and MixNibbles, the difference will be propagated to 4 bytes either in the left or the right 4-byte tuple. Thus Δ_2 will have minimum 4 active S-boxes in the second round. For simplicity, we assume L_2 contains 4 active bytes while R_2 remains zero. After the RotateNibbles step, both the left and the right tuples will have 2 active bytes. Since the branch number of MixNibbles is 5, the minimum number of active bytes with the differential characteristic Δ_3 will be 6. After RotateNibbles in the third round, if the active bytes in L_3 is 2 or 3, R_3 will have 4 or 3 active bytes respectively. In either of the situations, Δ_4 will have minimum 3+1 or 2+2 active bytes. In this general case, the minimum number of active S-boxes after four rounds is $1 + 4 + 6 + 4 = 15$.
- If there is 2 non-zero bytes in L_1, the difference will be propagated to at least 3 bytes after MixNibbles. For simplicity, we assume that L_2 contains 3 active bytes while R_2 remains zero. After RotateNibbles, the active bytes in L_2 is 1 or 2, while R_2 will have 2 or 1. Since the branch number of MixNibbles is 5, the minimum number of active bytes with the differential characteristic Δ_3 will be 7. After RotateNibbles in the third round, if the active bytes in L_3 is 3 or 4, R_3 will have 4 or 3 active bytes respectively. In either of the situations, Δ_4 will have minimum 2+1 or 1+2 active bytes. In this general case, the minimum number of active S-boxes after four rounds is $2 + 3 + 7 + 3 = 15$.
- In the case of 3 (or 4) active bytes in L_1, the difference patterns of the first three rounds will be identical to the case of 2 active bytes in the last three rounds. The minimum active bytes after three rounds are $3+7+3$ (or $4+6+4$). In the third round, first we choose all 3 (or 4) active bytes to be moved to L_3 after RotateNibbles. After MixNibbles, the active bytes will be at least 2 (or 1) since the branch number is 5. In any other choice, the active bytes in the forth round will be no less than 2 (or 1). Thus the minimum number of active S-boxes after four rounds is $3+7+3+2 = 15$ (or $4 + 6 + 4 + 1 = 15$) in this general case.

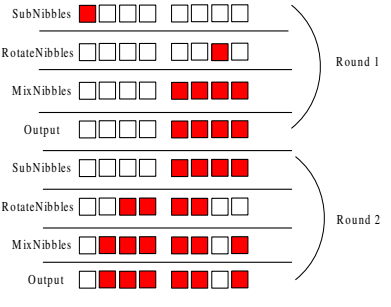

Fig. 4. A typical differential pattern of KLEIN

Figure 4 describes a typical differential pattern of KLEIN, where the colored boxes denote the active bytes in a trial. Without loss of generality, the same differential patterns will be followed where all active bytes are in R_1. If both L_1 and R_1 have one

or more active bytes, it is straightforward that the minimum number of active S-boxes will be no less than 15. Thus any four-round differential characteristic of KLEIN has a minimum of 15 active S-boxes. □

In RotateNibbles and MixNibbles, if we choose operations over $GF(2^4)$ for the MDS code, the active S-boxes in a four-round differential characteristic can be lower bounded by 25. Although a higher number of active S-boxes means KLEIN can use less rounds to be secure, our tuning experiments in sensors show that the software performance will be sacrificed by operations over $GF(2^4)$ (e.g. bit-shifting from leftmost to rightmost). However, it always requires a trade-off between the performance and the security. For ultra-lightweight in hardware, any differential characteristic of PRESENT has only 10 active S-boxes after 5 rounds.

In Rijndael, the coefficients of the MixColumns step are selected for assuring that both the differential branch number and the linear branch number are equal to 5. Based on the combination of RotateNibbles and MixNibbles, KLEIN also has the same property on the branch numbers. Therefore the minimum number of active S-boxes in a four-round linear approximation can be derived from the four-round differential propagation result of KLEIN. For brevity, the proof is omitted here.

Theorem 2. *Any four-round linear approximation of KLEIN has a minimum of 15 active S-boxes.*

The strength of a cipher against differential attacks is reflected by the maximum probability of differential , i.e. a collection of characteristics. However, in cryptanalysis we often assume that one characteristic has a much larger probability than the other characteristics of the differential. Thus a characteristic with the maximum probability is taken as an estimate of the probability of the differential. Similar assumptions can be found in linear attacks as well. Based on the minimum active S-boxes of characteristics in certain rounds, we can also derive the resistance of the differential and linear attacks on KLEIN. Since any differential characteristic over the KLEIN S-box has a maximum 2^{-2} possibility, the security against differential attacks of KLEIN-64 can be estimated as follows.

Lemma 1. *Let ϵ^d_{12R} be the maximum probability of a differential characteristic of 12 rounds of KLEIN-64. Then $\epsilon^d_{12R} \leq (2^{-2})^{12\times 15/4} \approx 2^{-90}$.*

Since any linear characteristic over the KLEIN S-box has a correlation 2^{-2}, the security against linear attacks of the full-round KLEIN-64 is described as follows.

Lemma 2. *Let ϵ^l_{12R} be the maximal bias of a linear approximation of 12 rounds of KLEIN-64. Then $\epsilon^l_{12R} \leq (2^{-2})^{12\times 15/4} \approx 2^{-90}$.*

By following the similar analysis, we note that the security of KLEIN-80/96 against linear and differential attacks can be gauged with 16/20 rounds.

Lemma 3. *Let ϵ^d_{16R} be the maximum probability of a differential characteristic of 16 rounds of KLEIN-80. Then $\epsilon^d_{16R} \leq (2^{-2})^{16\times 15/4} \approx 2^{-120}$.*

Lemma 4. *Let ϵ^l_{16R} be the maximum bias of a linear approximation of 16 rounds of KLEIN-80. Then $\epsilon^l_{16R} \leq (2^{-2})^{16\times 16/4} \approx 2^{-120}$.*

Lemma 5. *Let ϵ^d_{20R} be the maximum probability of a differential characteristic of 20 rounds of KLEIN-96. Then $\epsilon^d_{20R} \leq (2^{-2})^{20\times 15/4} \approx 2^{-150}$.*

Lemma 6. *Let ϵ^l_{20R} be the maximum bias of a linear approximation of 20 rounds of KLEIN-96. Then $\epsilon^l_{20R} \leq (2^{-2})^{20\times 15/4} \approx 2^{-150}$.*

Based on the above results, the KLEIN family of ciphers have a good security margin in the full rounds. The extra security margin of a block cipher may benefit the lightweight design of block-cipher-based hash functions or message authentication codes [15,21].

3.2 Key Schedule Attacks

Since there are no established guidelines for the design of key schedules, both a wide variety of designs and a wide variety of schedule-specific attacks have been proposed. The most effective attacks come under the general heading of related-key attacks and slide attacks, and both rely on the build-up of identifiable relationships between different sets of subkeys. To counter this threat, we use a round-dependent counter so that the subkey sets cannot easily be symmetric. We also use the same KLEIN S-box to provide the non-linearity of the subkeys whilst saving the implementation costs. For related-key attacks, we have the following properties for protection.

- For KLEIN-64/80/96, each bit in the key register depends on at least 4 user-supplied bits after 4/5/6 rounds.
- For KLEIN-64/80/96, all the bits in the key register are a non-linear function of the 64/80/96-bit user-supplied key by 8/10/12 rounds.

3.3 Integral Attack

Integral cryptanalysis are usually applied to exploit vulnerabilities in byte-oriented block ciphers, such as AES [14]. An integral attack will investigate the propagation of sums of many values, whilst a differential attack will consider the propagation of differences between pairs. In a byte-oriented cipher, the sum of a group differences might be a predictable value after certain rounds. For better software performance, the design of KLEIN also adopts a byte-oriented structure like AES. Thus it also faces a similar vulnerability on integral attacks [30].

First we consider a five-round integral attack on KLEIN, which is based on the attack given by Knudsen and Wagner on AES [30]. The attacker chooses a group of 256 plaintexts, which have equal values in all bytes except one. According to the mathematic properties of the RotateNibbles and MixNibbles steps, the sum of 256 bytes will be zero after three rounds of encryption. Then the attacker will guess 4 key bytes in the fourth round and 1 key bytes in the fifth round. If the $4 + 1 = 5$ key bytes are right, the sum of all 256 values should also be zero after five rounds. For KLEIN-80/96, we can extend the above attack to six rounds. Thus we need a collection of 2^{32} plaintexts in the first round and guess $4 + 5 = 9$ bytes in total. To the best of our knowledge, any integral attack on KLEIN over eight rounds will be more complicated than exhaustive key searches or requires the complete code book. Except byte-oriented integral

attacks, One could try nibble-oriented attacks with 16 plaintexts which have equal values except one nibble. However, since the MixNibbles step is fully based on multiplications in $GF(2^8)$, the sum of 16 nibbles is unpredictable after three rounds. Therefore nibble-oriented integral attacks will not be more feasible than byte-oriented ones.

3.4 Algebraic Attack

The algebraic attack as well as the *Cube Attack* [17], requires the algebraic form describing the output bits has a relatively small degree in terms of the input bits being processed. To exploit the algebraic relations between input and output bits of a block cipher, attackers may consider a subset of input bits whilst leave the other fixed. In the S-box of KLEIN, every output bit can be represented by a 3-degree polynomial with 4 input variables in ANF. For total 64 input bits, the complexity of finding the polynomials for the entire cipher soon becomes too large. In the full-round KLEIN-64, the number of S-boxes in the encryption and the key schedule equals $n = 12 \times 16 + 12 \times 4 = 240$. It is well-known that any 4-bit S-box can be represented by at least 21 quadratic equations over $GF(2)$. Thus in KLEIN-64, we have the number of quadratic equations $n \times 21 = 5040$ in $n \times 8 = 1920$ variables. By changing the number of rounds, similar results can easily be extended to KLEIN-80 and KLEIN-96. In our experiment, we were unable to transform three-round KLEIN-64 to the ANF equations in a reasonable time.

3.5 Side-Channel Attack

Since it is easy to add noises and loops in the software implementation of a block cipher to avoid side-channel attacks, here we will discuss on how to secure the hardware implementation of KLEIN. Except for the SubNibbles step, KLEIN is completely linear. The S-box of KLEIN can be implemented to resist side-channel attacks even in the presence of glitches using the secret sharing method proposed by Nikova *et al.* [41]. Also the linear part of KLEIN can be securely processed by using independent shares. A survey of lightweight cryptography and DPA countermeasures [39] estimates that the masking based on secret sharing will increase the hardware overhead with a factor of 3, which is still promising because it has a moderate area overhead and was theoretically proven to be secure against DPA attacks [41].

4 Performance

Here we analyze the performance of KLEIN. Based on legacy low-resource sensors TelosB (with 16-bit TI MSP430 microcontroller) and IRIS with (8-bit ATmega128L microcontroller), a detailed comparison of KLEIN and other related ciphers is given in Table 2. These two platforms are chosen because of their opposing characteristics: TelosB has more RAM than IRIS (10 KB vs 8 KB) but IRIS has a larger Flash memory than TelosB (128 KB vs 48 KB); TelosB's transceiver CC2420 supports hardware AES encryption but IRIS does not. For resource-constrained devices, a lower RAM cost would be beneficial for power consumptions and manufactory expenses. Since the

CC2420 chip on TelosB also supports AES hardware encryption, we also test its performance by implementing the standalone AES encryption of CC2420 [53]. The result shows AES hardware implementation has a great improvement on RAM and ROM costs, while the processing speed is even lower than the software implementation. We attribute this latency to the fact that the hardware AES encryption function must power up the CC2420 chip on TelosB in advance. The latency might be improved by setting CC2420 in the standby mode, but at the expense of increased power consumptions.

From a wide range of block ciphers, PRESENT is chosen because it is ultra-lightweight for highly resource-constrained [4], while Skipjack is proven to be software-efficient for 8-bit processors [32]. Both of them have similar block and key sizes as KLEIN. For Hummingbird encryption, Engels *et al.* [18] shows that the speed optimized implementation on 8-bit microcontrollers is about 28.9% slower than PRESENT encryption when the message length is 64 bits. On 16-bit microcontrollers Hummingbird achieves around $50\% \sim 78\%$ performance improvements for different message blocks [18]. The KATAN family ciphers and PRINTCipher, while space-efficient for hardware implementation, are problematic in software performance. This is because KATAN ciphers utilize bit manipulations extensively, whereas PRINTCipher operates on 3 bits at a time, which is at odds with existing 8/16/32-bit architectures.

The ciphers are written in nesC for the TinyOS 2.1.1 platform. This version of TinyOS does provide support for CC2420's AES encryption, but only in conjunction with radio operations. Support for the so-called standalone AES encryption is provided by Zhu's code [53]. The default optimization strategy of TinyOS ("-Os", which minimizes size but not at the expense of speed) is applied to all ciphers except software AES and DESXL on IRIS. For these exceptions, "-O1" and "-O0" are used respectively to bypass compilation errors. The maximum stack size on TelosB is measured using MSPsim [19]. There is no known tool for the same purpose on IRIS. The processing speeds are measured in encryption/decryption, whilst the storage costs are calculated in together. Since there is no such an instruction in TelosB or IRIS which can measure the processing speed by cycles per byte, the speed is compared by using the results of processing the same 16-byte message in milliseconds. All software implementations are optimized using look-up tables. On IRIS, the tables are specifically programmed to be stored in Flash memory. The performance comparison in Table 2 shows that KLEIN is competitive for low-resource applications and especially suitable for sensors. Although the block processing speeds of KLEIN are lower than Skipjack, it is a reasonable trade-off considering the security margin of KLEIN.

In KLEIN, the MixNibbles step will be a non-straightforward part of hardware design. Since it is the same as the MixColumns step of AES, we may simply borrow the idea from AES hardware implementations. Feldhofer *et al.* [20] shows a hardware-efficient implementation of the MixColumns step, which only costs about 340 GE with a 32-bit width. Because the MixNibbles step can be paralleled by two 32-bit tuples, Feldhofer *et al.*'s implementation can also be used in KLEIN families. The RotateNibbles step is simple byte-shift operations, which can be implemented with a minimum hardware.

For a hardware implementation, a low-cost RFID tag might have between 1,000 and 10,000 GE in total, while its security components may occupy up to 2,000 GE only

Table 2. The software performance of KLEIN and related block ciphers

	Performance on IRIS				
Algorithm	Key length (bit)	Block size (bit)	RAM (byte)	ROM (byte)	Processing speed (ms per 16-byte message)
AES-128 (software implementation)	128	128	295	14216	1.32/1.29
NOEKEON (indirect encryption)	128	128	111	4472	3.33/3.33
NOEKEON (direct encryption)	128	128	111	4424	3.33/3.33
DESXL	184	64	306	32186	6.43/5.68
GOST	256	64	233	14342	1.93/1.89
SEA	96	96	249	1904	3.76/3.67
HIGHT	128	64	117	2510	1.84/1.46
Hummingbird	128	16	159	2646	4.30/9.14
PRESENT-80	80	64	365	6866	4.06/9.34
PRINTCipher-48	80	48	125	6184	21.6/14.9
KATAN-64	80	64	625	3260	80.5/43.5
mCrypton-64	64	64	355	9768	5.20/4.41
mCrypton-96	96	64	355	10252	5.37/4.41
mCrypton-128	128	64	355	11160	5.49/4.51
Skipjack	80	64	133	2566	0.90/0.90
KLEIN-64	64	64	105	2582	0.96/1.25
KLEIN-80	80	64	107	2672	1.21/1.70
KLEIN-96	96	64	109	2782	1.52/2.11

	Performance on TelosB					
Algorithm	Key length (bit)	Block size (bit)	RAM (byte)	ROM (byte)	Max stack (byte)	Processing speed (ms per 16-byte message)
AES-128 (software implementation)	128	128	218	10898	230	1.71/1.68
AES-128 (hardware encryption)	128	128	60	900	116	2.29
NOEKEON (indirect encryption)	128	128	56	3544	258	2.38/2.43
NOEKEON (direct encryption)	128	128	56	4224	242	2.38/2.42
DESXL	184	64	186	6966	144	1.86/1.86
GOST	256	64	190	4748	120	2.56/2.52
SEA	96	96	204	2754	120	7.3/6.89
HIGHT	128	64	40	2050	132	2.15/2.15
Hummingbird	128	16	82	1822	116	4.61/9.69
PRESENT-80	80	64	288	6424	128	6.6/11.1
PRINTCipher-48	80	48	48	6210	128	28.3/21.3
KATAN-64	80	64	548	2628	202	120/121
mCrypton-64	64	64	248	7816	182	3.4/3.91
mCrypton-96	96	64	248	8026	158	3.4/3.91
mCrypton-128	128	64	248	8748	158	3.4/3.91
Skipjack	80	64	56	1542	130	1.37/1.33
KLEIN-64	64	64	50	2980	186	1.97/2.5
KLEIN-80	80	64	52	3112	178	2.62/3.4
KLEIN-96	96	64	54	3266	182	3.32/4.55

[26,47]. The area restriction could be looser on sensors. By using VeriSilicon GSMC 0.13um low-power process high-density standard cell library, Yalcin *e*t al. implemented KLEIN-64 for an ultra-lightweight crypto processor [2]. Based on their results and synthesized with TSMC 0.18 μm Process 1.8-Volt SAGE-X Standard Cell Library, the hardware implementation results of KLEIN families are compared to the related block ciphers in Table 3.

Table 3. Hardware implementation results comparison of KLEIN and related block ciphers

	Hardware Encryption				
Algorithm	Implementation	Logic process (μm)	Datapath (bits)	Area in GE	Cycle per block
AES-128	[51]	0.11	32	5400	54
	[24]	0.13	8	3100	160
	[40]	0.18	8	2400	226
HIGHT	[22]	0.25	64	3048	34
PRESENT-80	[4]	0.18	64	1570	32
			4	1075	563
KLEIN-64	[2]	0.13	8	1365	96
KLEIN-64	this paper	0.18	64	2475	13
			8	1397	103
			4	1220	207
KLEIN-80	this paper	0.18	64	2629	17
			8	1630	135
			4	1478	271
KLEIN-96	this paper	0.18	64	2769	21
			8	1696	167
			4	1528	335

5 Conclusion

In this paper, we have proposed a new family of block ciphers called KLEIN. The goal of our design is to provide a practical and secure cipher for low-resource applications, especially for RFIDs and wireless sensor networks. Although KLEIN mainly focuses on software implementations, it also enjoys hardware efficiency resulting from its simple structure with an involutive S-box. The various key lengths of KLEIN offer a flexibility and a moderate security level for ubiquitous applications. Therefore, our design increases the available options for lightweight block ciphers in low-resource applications.

Acknowledgement. We would like to thank Begül Bilgin for her helpful results on the hardware implementations of KLEIN. And we also thank Hongjun Wu and many anonymous reviewers for their valuable comments. We are grateful to Xinxin Fan, Chae Hoon Lim, Axel Poschmann, Eric Smith, Francois-Xavier Standaert for sharing their test vectors and reference source code with us. The work described in this paper has been supported [in part] by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II. The authors acknowledge the financial support of SenterNovem for the ALwEN project, grant PNE07007. Yee Wei Law is partly supported by the Australian Research Council under contract number DP1095452, and the European Commission under contract number FP7-257992 (SmartSantander). The authors also thank the support of NSFC Grant 61100201.

References

1. 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)
2. Bilgin, B., Kavun, E.B., Yalcin, T.: Towards an Ultra Lightweight Crypto Processor. In: Workshop on Lightweight Security & Privacy: Devices, Protocols, and Applications (Lightsec 2011), pp. 76–83. IEEE CS, Los Alamitos (2011)
3. Bogdanov, A., Knudsen, L.R., 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)
4. Bogdanov, A., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y.: Hash Functions and RFID tags: Mind the Gap. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 283–299. Springer, Heidelberg (2008)
5. 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)
6. 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)
7. Chipcon: CC2420: 2.4 GHz IEEE 802.15.4/Zigbee-ready RF transceiver, `http://focus.ti.com/lit/ds/symlink/cc2420.pdf`
8. Cho, J.Y.: Linear Cryptanalysis of Reduced-Round PRESENT. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 302–317. Springer, Heidelberg (2010)
9. Collard, B., Standaert, F.-X.: A Statistical Saturation Attack Against The Block Cipher PRESENT. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 195–210. Springer, Heidelberg (2009)
10. Crossbow: IRIS wireless measurement system, `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/IRIS_Datasheet.pdf`
11. Crossbow: TelosB mote platform, `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf`
12. Daemen, J., Knudsen, L.R., Rijmen, V.: Linear Frameworks for Block Ciphers. Designs, Codes and Cryptography 22(1), 65–87 (2001)
13. Daemen, J., Peeters, M., Van Assche, G., Rijmen, V.: The NOEKEON Block Cipher. The NESSIE Proposal (2000)
14. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, Heidelberg (2002)
15. Daemen, J., Rijmen, V.: A New MAC Construction ALRED and A Specific Instance ALPHA-MAC. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 1–17. Springer, Heidelberg (2005)
16. Daemen, J., Rijmen, V.: New Criteria for Linear Maps in AES-Like Ciphers. Cryptography and Communications 1(1), 47–69 (2009)
17. Dinur, I., Shamir, A.: Cube Attacks on Tweakable Black Box Polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009)
18. Engels, D., Fan, X., Gong, G., Hu, H., Smith, E.M.: Hummingbird: Ultra-Lightweight Cryptography for Resource-Constrained Devices. In: Sion, R., Curtmola, R., Dietrich, S., Kiayias, A., Miret, J.M., Sako, K., Sebé, F. (eds.) RLCPS, WECSR, and WLC 2010. LNCS, vol. 6054, pp. 3–18. Springer, Heidelberg (2010)

19. Eriksson, J., Dunkels, A., Finne, N., Österlind, F., Voigt, T.: MSPsim - An Extensible Simulator for MSP430-Equipped Sensor Boards. In: Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo Session, Delft, The Netherlands (January 2007)
20. Feldhofer, M., Wolkerstorfer, J., Rijmen, V.: AES Implementation on a Grain of Sand. IEEE Proceedings on Information Security 152(1), 13–20 (2005)
21. Gong, Z., Hartel, P., Nikova, S., Zhu, B.: Towards Secure and Practical MACs for Body Sensor Networks. In: Roy, B.K., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 182–198. Springer, Heidelberg (2009)
22. 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)
23. Granboulan, L.: Flaws in Differential Cryptanalysis of Skipjack. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 328–335. Springer, Heidelberg (2002)
24. Hamalainen, P., Alho, T., Hannikainen, M., Hamalainen, T.D.: Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core. In: DSD 2006: Proceedings of the 9th EUROMICRO Conference on Digital System Design, pp. 577–583. IEEE Computer Society, Washington, DC, USA (2006)
25. Healy, M., Newe, T., Lewis, E.: Analysis of Hardware Encryption Versus Software Encryption on Wireless Sensor Network Motes. In: Mukhopadhyay, S.C., Gupta, G.S. (eds.) Smart Sensors and Sensing Technology 2008. LNEE, vol. 20, pp. 3–14. Springer, Heidelberg (2008)
26. Juels, A., Weis, S.A.: Authenticating Pervasive Devices with Human Protocols. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 293–308. Springer, Heidelberg (2005)
27. Knudsen, L.R., Leander, G., Poschmann, A., Robshaw, M.J.B.: PRINTCipher: A Block Cipher for IC-Printing. In: Mangard and Standaert [37], pp. 16–32
28. Knudsen, L.R., Raddum, H.: On NOEKEON. The NESSIE Report (April 2001)
29. Knudsen, L.R., Robshaw, M.J.B., Wagner, D.: Truncated Differentials and Skipjack. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 165–180. Springer, Heidelberg (1999)
30. Knudsen, L.R., Wagner, D.: Integral Cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002)
31. Könighofer, R.: A Fast and Cache-Timing Resistant Implementation of the AES. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 187–202. Springer, Heidelberg (2008)
32. Law, Y.W., Doumen, J., Hartel, P.H.: Survey and Benchmark of Block Ciphers for Wireless Sensor Networks. ACM Trans. Sen. Netw. 2(1), 65–93 (2006)
33. 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)
34. 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)
35. Lim, C.H., Korkishko, T.: mCrypton – A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors. In: Song, J., Kwon, T., Yung, M. (eds.) WISA 2005. LNCS, vol. 3786, pp. 243–258. Springer, Heidelberg (2006)
36. Mangard, S., Popp, T., Gammel, B.M.: Side-Channel Leakage of Masked CMOS Gates. In: Menezes, A.J. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 351–365. Springer, Heidelberg (2005)
37. Mangard, S., Standaert, F.X. (eds.): CHES 2010. LNCS, vol. 6225. Springer, Heidelberg (2010)
38. Matsui, M.: New Structure of Block Ciphers with Provable Security against Differential and Linear Cryptanalysis. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 205–218. Springer, Heidelberg (1996)

39. Moradi, A., Poschmann, A.: Lightweight Cryptography and DPA Countermeasures: A Survey. In: Sion, R., Curtmola, R., Dietrich, S., Kiayias, A., Miret, J.M., Sako, K., Sebé, F. (eds.) RLCPS, WECSR, and WLC 2010. LNCS, vol. 6054, pp. 68–79. Springer, Heidelberg (2010)
40. 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)
41. Nikova, S., Rijmen, V., Schläffer, M.: Secure Hardware Implementation of Non-Linear 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)
42. NIST. Skipjack and KEA algorithm Specifications (version 2.0). NIST online document (May 1998), http://csrc.nist.gov/groups/ST/toolkit/documents/skipjack/skipjack.pdf
43. Ohkuma, K.: Weak Keys of Reduced-Round PRESENT for Linear Cryptanalysis, pp. 249–265. Springer, Heidelberg (2009)
44. Özen, O., Varıcı, K., Tezcan, C., Kocair, Ç.: Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 90–107. Springer, Heidelberg (2009)
45. Paar, C., Poschmann, A., Robshaw, M.: New Designs in Lightweight Symmetric Encryption. In: Kitsos, P., Zhang, Y. (eds.) RFID Security: Techniques, Protocols and System-on-Chip Design, pp. 349–371. Springer, Heidelberg (2008)
46. Phan, R.C.W.: Cryptanalysis of Full Skipjack Block Cipher. Electronic Letters, 69–71 (2002)
47. Poschmann, A.: Lightweight Cryptography - Cryptographic Engineering for a Pervasive-World. PhD thesis, Ruhr-University Bochum, Germany (2009)
48. Poschmann, A., Ling, S., Wang, H.: 256 Bit Standardized Crypto for 650 Ge - Gost Revisited. In: Mangard and Standaert [37], pp. 219–233
49. 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)
50. Saarinen, M.J.O.: Cryptanalysis of Hummingbird-1. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 328–341. Springer, Heidelberg (2011)
51. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A Compact Rijndael Hardware Architecture with S-Box Optimization. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 239–254. Springer, Heidelberg (2001)
52. Standaert, F.X., Piret, G., Gershenfeld, N., Quisquater, J.J.: SEA: A Scalable Encryption Algorithm for Small Embedded Applications. In: Domingo-Ferrer, J., Posegga, J., Schreckling, D. (eds.) CARDIS 2006. LNCS, vol. 3928, pp. 222–236. Springer, Heidelberg (2006)
53. Zhu, B.: The Standalone AES Encryption of CC2420 (TinyOS 2.10 and MICAz) (December 2008), http://cis.sjtu.edu.cn/index.php/Bo_Zhu

Appendix A. Test Vectors of KLEIN

Table 4. Test vectors for KLEIN-64

Key	Message	Cipher
0000 0000 0000 0000	FFFF FFFF FFFF FFFF	CDC0 B51F 1472 2BBE
FFFF FFFF FFFF FFFF	0000 0000 0000 0000	6456 764E 8602 E154
1234 5678 90AB CDEF	FFFF FFFF FFFF FFFF	5923 56C4 9971 76C8
0000 0000 0000 0000	1234 5678 90AB CDEF	629F 9D6D FF95 800E

Table 5. Test vectors for KLEIN-80

Key	Message	Cipher
0000 0000 0000 0000 0000	FFFF FFFF FFFF FFFF	6677 E20D 1A53 A431
FFFF FFFF FFFF FFFF FFFF	0000 0000 0000 0000	8224 7502 273D CC5F
1234 5678 90AB CDEF 1234	FFFF FFFF FFFF FFFF	3F21 0F67 CB23 687A
0000 0000 0000 0000 0000	1234 5678 90AB CDEF	BA52 39E9 3E78 4366

Table 6. Test vectors for KLEIN-96

Key	Message	Cipher
0000 0000 0000 0000 0000 0000	FFFF FFFF FFFF FFFF	DB9F A7D3 3D8E 8E36
FFFF FFFF FFFF FFFF FFFF FFFF	0000 0000 0000 0000	15A3 A033 86A7 FEC6
1234 5678 90AB CDEF 1234 5678	FFFF FFFF FFFF FFFF	7968 7798 AFDA 0BC3
0000 0000 0000 0000 0000 0000	1234 5678 90AB CDEF	5006 A987 A500 BFDD