KATAN & KTANTAN — A Family of Small and Efficient Hardware-Oriented Block Ciphers

Christophe De Cannière[1] and Orr Dunkelman[1,2,⋆] Miroslav Knežević[1,⋆⋆]

[1] Katholieke Universiteit Leuven
Department of Electrical Engineering ESAT/SCD-COSIC
and
Interdisciplinary Center for Broad Band Technologies
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
{christophe.decanniere,miroslav.knezevic}@esat.kuleuven.be

[2] École Normale Supérieure
Département d'Informatique,
CNRS, INRIA
45 rue d'Ulm, 75230 Paris, France.
orr.dunkelman@di.ens.fr

Abstract. In this paper we propose a new family of very efficient hardware oriented block ciphers. The family contains six block ciphers divided into two flavors. All block ciphers share the 80-bit key size and security level. The first flavor, KATAN, is composed of three block ciphers, with 32, 48, or 64-bit block size. The second flavor, KTANTAN, contains the other three ciphers with the same block sizes, and is more compact in hardware, as the key is burnt into the device (and cannot be changed).
The smallest cipher of the entire family, KTANTAN32, can be implemented in 462 GE while achieving encryption speed of 12.5 KBit/sec (at 100 KHz). KTANTAN48, which is the version we recommend for RFID tags uses 588 GE, whereas KATAN64, the largest and most flexible candidate of the family, uses 1054 GE and has a throughput of 25.1 Kbit/sec (at 100 KHz).

1 Introduction

Low-end devices, such as RFID tags, are deployed in increasing numbers each and every day. Such devices are used in many applications and environments, leading to an ever increasing need to provide security (and privacy). In order to satisfy these needs, several suitable building blocks, such as secure block ciphers, have to be developed.

The problem of providing secure primitives in these devices is the extremely constrained environment. The primitive has to have a small footprint (where any additional gate might lead to the solution not being used), reduced power consumption (as these devices either rely on a battery or on an external electromagnetic field to supply them the required energy), and with sufficient speed (to allow the use of the primitive in real protocols).

The raising importance as well as the lack of secure and suitable candidates, has initiated a research aiming to satisfy these requirements. The first candidate block cipher for these devices was the DESL algorithm [19]. DESL is based on the general structure of DES, while using a specially selected S-box. DESL has key size of 56 bits and a

⋆ This author was partially supported by the France Telecom Chaire.
⋆⋆ This author is supported in part by IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

footprint of 1848 GE. The second candidate for the mission is the PRESENT block cipher [4]. PRESENT has an SP-Network structure, and it can be implemented using the equivalent of 1570 GE. A more dedicated implementation of PRESENT in 0.35μm CMOS technology reaches 1000 GE [20].[1] The same design in 0.25μm and 0.18μm CMOS technology consumes 1169 and 1075 GE, respectively.

Some stream ciphers, such as grain [11] and trivium [6] may also be considered fit for these constrained environments, with 1293 and 749 GE[2] implementations, respectively. However, some protocols cannot be realized using stream ciphers, thus, leaving the issue of finding a more compact and secure block cipher open.

In this paper we propose a new family of block ciphers composed of two sets. The first set of ciphers is the KATAN ciphers, KATAN32, KATAN48 and KATAN64. All three ciphers accept 80-bit keys, and have a different block size (n-bit for KATANn). These three block ciphers are highly compact and achieve the minimal size (while offering adequate security). The second set, composed of KTANTAN32, KTANTAN48, and KTANTAN64, realize even smaller block ciphers in exchange for agility. KTANTANn is more compact than KATANn, but at the same time, is suitable only for cases where the device is initialized with one key that can never be altered, i.e., for the KTANTAN families, the key of the device is burnt into the device. Thus, the only algorithmic difference between KATANn and KTANTANn is the key schedule (which may be considered slightly more secure in the KATANn case).

While in this paper we put emphasis on the smallest possible variants, it can be easily seen that increasing the speed of the implementation is feasible with only a small hardware overhead. Therefore, we provide more implementation results in Appendix B. We implemented all six ciphers of the family using an *fsc0l_d_sc_tc* 0.13μm family standard cell library tailored for UMC's 0.13μm Low Leakage process. We compare our results with previous constructions in Table 1. We note here that some of the implementations achieve an amazingly low gate count due to the number of GE per bit of memory used. This is the issue inherent not only to the encryption algorithm, but also a matter of the technology that is used. Thus, we give a detailed explanation addressing the possible bit representation.

We organize this paper as follows: In Section 2 we describe the design criteria used in the construction of the KATAN family. Section 3 presents the building blocks used in our construction as well as the implementation issues related to them. In Sections 4 and 5 we present the KATAN and the KTANTAN families, respectively. The security analysis results are given in Section 6. Several compartive tradeoffs concerning the implemtnation speed and size of the KATAN and KTANTAN families are reported in Appendix B. Finally, we summarize our results in Section 7.

2 Motivation and Design Goals

Our main design goal was to develop a secure 80-bit block cipher with as minimal number of gates as possible. Such ciphers are needed in many constrained environments, e.g., RFID tags and sensor networks.

[1] Comparing to 0.13μm CMOS technology we note here that the physical size of the chip (in μm^2) is about 8 times bigger than the design with the same number of gate equivalents in 0.13μm CMOS technology.

[2] This work is a full-custom design implemented with C^{2}MOS dynamic logic [16]. The die size is equivalent to 749 standard CMOS logic NAND gates. The clock frequency required for this solution is far from being suitable for constrained environments.

Table 1. Comparison of Ciphers Designed for Low-End Environments (optimized for size).

Cipher	Block (bits)	Key (bits)	Size (GE)	Gates per Memory Bit	Throughput[⋆] (Kb/s)	Logic Process
AES-128 [8]	128	128	3400	7.97	12.4	0.35 μm
AES-128 [10]	128	128	3100	5.8	0.08	0.13 μm
HIGHT [12]	64	128	3048	N/A	188.25	0.25 μm
mCrypton [15]	64	64	2420	5	492.3	0.13 μm
DES [19]	64	56	2309[†]	12.19	44.4	0.18 μm
DESL [19]	64	56	1848[†]	12.19	44.4	0.18 μm
PRESENT-80 [4]	64	80	1570	6	200	0.18 μm
PRESENT-80 [20]	64	80	1000	N/A	11.4	0.35 μm
Grain [9]	1	80	1294	7.25	100	0.13 μm
Trivium [16]	1	80	749	2[◊]	100[‡]	0.35 μm
KATAN32	32	80	802	6.25	12.5	0.13 μm
KATAN48	48	80	927	6.25	18.8	0.13 μm
KATAN64	64	80	1054	6.25	25.1	0.13 μm
KTANTAN32	32	80	462	6.25	12.5	0.13 μm
KTANTAN48	48	80	588	6.25	18.8	0.13 μm
KTANTAN64	64	80	688	6.25	25.1	0.13 μm

⋆ — A throughput is estimated for frequency of 100 KHz.
† — Fully serialized implementation (the rest are only synthesized).
‡ — This throughput is projected, as the chip requires higher frequencies.
◊ — This is a full-custom design using C^2MOS dynamic logic.

While analyzing the previous solutions to the problem, we have noticed that the more compact the cipher is, a larger ratio of the area is dedicated for storing the intermediate values and key bits. For example, in grain [11], almost all of the 1294 gates which are required, are used for maintaining the internal state. This phenomena also exist in DESL [19] and PRESENT [4], but to a lesser degree. This follows two-fold reasoning: First, stream ciphers need an internal state of at least twice the security level while block ciphers are exempt from this requirement. Second, while in stream ciphers it is possible to use relatively compact highly nonlinearity combining function, in block ciphers the use of S-box puts a burden on the hardware requirements.

Another interesting issue that we have encountered during the analysis of previous results is the fact that various implementations not only differ in the basic gate technology, but also in the number of gate equivalents required for storing a bit. In the standard library we have used in this work, a simple flip-flop implementation can take between 5 and 12 GE. This, of course, depends on the type of the flip-flop that is used (scan or standard D flip-flop, with or without set/reset signals, input and output capacitance, etc). Typical flip-flops that are used to replace a combination of a multiplexer and a flip-flop are, so called, scan flip-flops of which the most compact version, in our library, has a size equivalent to 6.25 GE. These flip-flops basically act as a combination of a simple D flip-flop and a MUX2to1. Use of this type of flip-flops is beneficial both for area and power consumption.

Here, we can notice that in PRESENT [4], the 80-bit key is stored in an area of about 480 GE, i.e., about 6 GE for one bit of memory, while in DESL, the 64-bit state is stored in 780 GE (about 12 GE for a single bit). As we have already discussed, this is related to many different factors such as the type of flip-flops, technology, library, etc. Finally, we note that in some cases (which do not necessarily fit an RFID tag due to practical reasons) it is possible to reduce the area required for storing one memory bit to only

8 transistors (i.e., about 2 GE) [16]. This approach achieves a much better comparison between different implementations, as usually changing the memory technology we can relatively easily counter the effects of implementers knowledge (or lack of), and discuss the true size of the proposed algorithm.

An additional issue which we observed is that in many low-end applications, the key is loaded once to the device and is never changed. In such instances, it should be possible to provide an encryption solution which can handle a key which is not stored in memory, preferably in a more efficient manner.

A final issue related to reducing the area requirements of the cipher is the block size. By decreasing the block size, it is possible to further reduce the memory complexity of the cipher. On the other hand, reducing the plaintext size to less than 32 bits has strong implications on the security of the systems using this cipher. For example, due to the birthday bound, a cipher with block size smaller than 32 bits is distinguishable from a family of random permutations after 2^{16} blocks.

The life span of a simple RFID tag indeed fits this restriction, but some RFID tags and several devices in sensor networks may need to encrypt larger amounts of data (especially if the used protocols require the encryption of several values in each execution). Thus, we decided to offer 3 block sizes to implementers — 32 bits, 48 bits, and 64 bits.

Our specific design goals are as follows:

– For an n-bit block size, no differential characteristic with probability greater than 2^{-n} exists for 128 rounds (about half the number of rounds of the cipher).
– For an n-bit block size, no linear approximation with bias greater than $2^{-n/2}$ exists for 128 rounds.
– No related-key key-recovery or slide attack with time complexity smaller than 2^{80} exists on the entire cipher.
– High enough algebraic degree for the equation describing half the cipher to thwart any algebraic attack.

We note that the first two conditions ensure that no differential-linear attack (or a boomerang attack) exist for the entire cipher as well. We also had to rank the possible design targets as follows:

– Minimize the size of the implementation.
– Keeping the critical path as short as possible.
– Increase the throughput of the implementation (as long as the increase in the foot print is small).
– Decrease the power consumption of the implementation.

3 General Construction and Building Blocks

Following the design of KeeLoq [17], we decided to adopt a cipher whose structure resembles a stream cipher. To this extent we have chosen a structure which resembles trivium [6], or more precisely, its two register variant bivium as the base for the block cipher. While the internal state of trivium was 288 bits to overcome the fact that each round, one bit of internal state is revealed, in the block cipher this extra security measure is unnecessary. Hence, we select the block size and the internal state of the cipher to be equal.

The structure of the KATAN and the KTANTAN ciphers is very simple — the plaintext is loaded into two registers (whose lengths depend on the block size). Each round, several bits are taken from the registers and enter two nonlinear Boolean functions. The output of the Boolean functions is loaded to the least significant bits of the registers (after they were shifted). Of course, this is done in an invertible manner. To ensure sufficient mixing, 254 rounds of the cipher are executed.

We have devised several mechanisms used to ensure the security of the cipher, while maintaining a small foot print. The first one is the use of an LFSR instead of a counter for counting the rounds and to stop the encryption after 254 rounds. As there are 254 rounds, an 8-bit LFSR with as sparse polynomial feedback can be used. The LFSR is initialized with some state, and the cipher has to stop running the moment the LFSR arrives to some predetermined state.

We have implemented the 8-bit LFSR counter, and the result fits a gate equivalent of 60 gates, while using an 8-bit counter (the standard alternative) took 80 gate equivalents. Moreover, the expected speed of the LFSR (i.e. the critical path) is shorter than the one for the 8-bit counter.

Another advantage for using LFSR is the fact that when considering one of the bits taken from it, we expect a sequence which keeps on alternating between 0's and 1's in a more irregular manner than in a counter (of course the change is linear). We use this feature to enhance the security of our block ciphers as we describe later.

One of the problems that may arise in such a simple construction is related to self-similarity attacks such as the slide attacks. For example, in KeeLoq [17] the key is used again and again. This made KeeLoq susceptible to several slide attacks (see for example [5, 13]). A simple solution to the problem is to have the key loaded into an LFSR with a primitive feedback polynomial (thus, altering the subkeys used in the cipher). This solution helps the KATAN family to achieve security against the slide attack.

While the above building block is suitable when the key is loaded into memory, in the KTANTAN family, it is less favorable (as the key is hardcoded in the device). Thus, the only means to prevent a slide attack is by generating a simple, non-repetitive sequence of bits from the key. To do so, we use the "round counter" LFSR, which produces easily computed bits, that at the same time follow a non-repetitive sequence.

The third building block which we use prevents the self-similarity attacks and increases the diffusion of the cipher. The cipher actually has two (very similar but distinct) round functions. The choice of the round function is made according to the most significant bit of the round-counting LFSR. This irregular update also increases the diffusion of the cipher, as the nonlinear update affects both the differential and the linear properties of the cipher.

Finally, both KATAN and KTANTAN were constructed such that an implementation of the 64-bit variants can support the 32-bit and the 48-bit variants at the cost of small extra controlling hardware. Moreover, given the fact that the only difference between a KATANn cipher and KTANTANn is the way the key is stored and the subkeys are derived, it is possible to design a very compact circuit that support all six ciphers.

4 The KATAN Set of Block Ciphers

The KATAN ciphers compose of three variants: KATAN32, KATAN48 and KATAN64. All the ciphers in the KATAN family share the key schedule which accepts an 80-bit key and 254 rounds as well as the use of the same nonlinear functions.

We start by describing KATAN32, and describe the differences for KATAN48 and KATAN64 later. KATAN32, the smallest of this family has a plaintext and ciphertext size of 32 bits. The plaintext is loaded into two registers L_1, and L_2 (of respective lengths of 13 and 19 bits) where the least significant bit of the plaintext is loaded to bit 0 of L_2, while the most significant bit of the plaintext is loaded to bit 12 of L_1. Each round, L_1 and L_2 are shifted to the left (bit i is shifted to position $i + 1$), where the new computed bits are loaded in the least significant bits of L_1 and L_2. After 254 rounds of the cipher, the contents of the registers are then exported as the ciphertext (where bit 0 of L_2 is the least significant of the ciphertext).

KATAN32 uses two nonlinear function $f_a(\cdot)$ and $f_b(\cdot)$ in each round. The nonlinear function f_a and f_b are defined as follows:

$$f_a(L_1) = L_1[x_1] \oplus L_1[x_2] \oplus (L_1[x_3] \cdot L_1[x_4]) \oplus (L_1[x_5] \cdot IR) \oplus k_a$$

$$f_b(L_2) = L_2[y_1] \oplus L_2[y_2] \oplus (L_2[y_3] \cdot L_2[y_4]) \oplus (L_2[y_5] \cdot L_2[y_6]) \oplus k_b$$

where IR is irregular update rule (i.e., $L_1[x_5]$ is XORed in the rounds where the irregular update is used), and k_a and k_b are the two subkey bits. For round i, k_a is defined to be k_{2i}, whereas k_b is k_{2i+1}. The selection of the bits $\{x_i\}$ and $\{y_j\}$ are defined for each variant independently, and listed in Table 2.

After the computation of the nonlinear functions, the registers L_1 and L_2 are shifted, where the MSB falls off (into the corresponding nonlinear function), and the LSB is loaded with the output of the second nonlinear function, i.e., after the round the LSB of L_1 is the output of f_b, and the LSB of L_2 is the output of f_a.

The key schedule of the KATAN32 cipher (and the other two variants KATAN48 and KATAN64) loads the 80-bit key into an LFSR (the least significant bit of the key is loaded to position 0 of the LFSR). Each round, positions 0 and 1 of the LFSR are generated as the round's subkey k_{2i} and k_{2i+1}, and the LFSR is clocked twice. The feedback polynomial that was chosen is a primitive polynomial with minimal hamming weight of 5 (there are no primitive polynomials of degree 80 with only 3 monomials):

$$x^{80} + x^{61} + x^{50} + x^{13} + 1.$$

We note that these locations compose a full difference set, and thus, are less likely to lead to a guess and determine attacks faster than exhaustive key search.

In other words, let the key be K, then the subkey of round i is $k_a||k_b = k_{2 \cdot i}||k_{2 \cdot i+1}$ where

$$k_i = \begin{cases} K_i & \text{for } i = 0 \dots 79 \\ k_{i-80} \oplus k_{i-61} \oplus k_{i-50} \oplus k_{i-13} & \text{Otherwise} \end{cases}$$

The differences between the various KATAN ciphers are:

– The plaintext/ciphertext size,
– The lengths of L_1 and L_2,
– The position of the bits which enter the nonlinear functions,
– The number of times the nonlinear functions are used in each round.

While the first difference is obvious, we define in Table 2 the lengths of the registers and the positions of the bits which enter the nonlinear functions used in the ciphers. The selection of the bits $\{x_i\}$ and $\{y_j\}$ are defined for each variant independently, and are listed in Table 2.

For KATAN48, in one round of the cipher the functions f_a and f_b are applied twice. The first pair of f_a and f_b is applied, and then after the update of the registers, they are applied again, using the same subkeys. Of course, an efficient implementation can implement these two steps in parallel. In KATAN64, each round applies f_a and f_b three times (again, with the same key bits).

We outline the structure of KATAN32 (which is similar to the round function of any of the KATAN variants or the KTANTAN variants) in Figure 1.

Finally, specification-wise, we define the counter which counts the number of rounds. The round-counter LFSR is initialized to the all 1's state, and clocked once using the

Cipher	$\lvert L_1 \rvert$	$\lvert L_2 \rvert$	x_1	x_2	x_3	x_4	x_5
KATAN32/KTANTAN32	13	19	12	7	8	5	3
KATAN48/KTANTAN48	19	29	18	12	15	7	6
KATAN64/KTANTAN64	25	39	24	15	20	11	9

Cipher	y_1	y_2	y_3	y_4	y_5	y_6
KATAN32/KTANTAN32	18	7	12	10	8	3
KATAN48/KTANTAN48	28	19	21	13	15	6
KATAN64/KTANTAN64	38	25	33	21	14	9

Table 2. Parameters defined for the KATAN family of ciphers

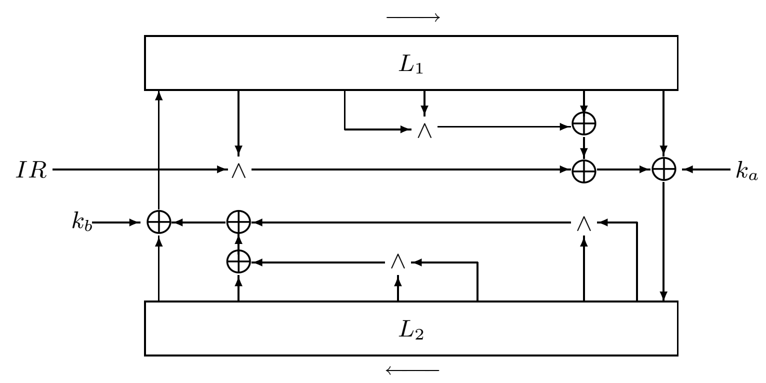

Fig. 1. The Outline of a round of the KATAN/KTANTAN ciphers

feedback polynomial $x^8 + x^7 + x^5 + x^3 + 1$. Then, the encryption process starts, and ends after 254 additional clocks when the LFSR returns to the all 1's state. As mentioned earlier, we use the most significant bit of the LFSR to control the irregular update (i.e., as the IR signal). For sake of completeness, in Table 3 in the Appendix we give the sequence of irregular rounds.

We note that due to the way the irregular update rule is chosen, there are no sequences of more than 7 rounds that share the pattern of the regular/irregular updates, this ensures that any self-similarity attack cannot utilize more than 7 rounds of the same function (even if the attacker chooses keys that suggest the same subkeys). Thus, it is easy to see that such attacks are expected to fail when applied to the KATAN family.

We implemented KATAN32 using Synopsys Design Compiler version Y-2006.06 and the *fsc0l_d_sc_tc* 0.13μm CMOS library. Our implementation requires 802 GE, of which 742 are used for the sequential logic, and 60 GE are used for the combinational logic. The power consumption at 100 KHz, and throughput of 12.5 Kbps is only 381 nW. This is a gate level power estimation obtained using Synopsys Design Compiler[3].

For KATAN48 the implementation size is 927 GE (of which 842 are for the sequential logic) and the total power consumption is estimated to 439 nW. For the 64-bit variant, KATAN64, the total area is 1054 GE (of which 935 are for the sequential logic) and the power consumption 555 nW.

Here we would like to note that the further area reduction for KATAN48 and KATAN64 is possible by utilizing a clock gating technique. As explained above, the only difference

[3] Although the gate level power estimation gives a rough estimate, it is useful for comparison with related work reported in the literature.

between KATAN32 on one hand and KATAN48 and KATAN64 on the other, is the number of nonlinear functions f_a and f_b applied with the same subkeys per single round. Therefore, we can clock the key register and the counter such that they are updated once in every two (three) cycles for KATAN48 (KATAN64). However, this approach reduces the throughput two (three) times respectively, and is useful only when the compact implementation is an ultimate goal. An area of 916 GE with the throughput of 9.4 Kb/s (at 100 KHz) is obtained for KATAN48 and 1027 GE with the throughput of 8.4 Kb/s (at 100 KHz) for KATAN64.

At the cost of little hardware overhead, a throughput of the KATAN family of block ciphers can be doubled or even tripled. To increase the speed of the cipher, we double (triple) the logic for the nonlinear functions f_a and f_b as well as the logic for the feedback coefficients of the counter and the key register. The implementation results are given in Appendix B.

5 The KTANTAN Family

The KTANTAN family is very similar to the KATAN family up to the key schedule (i.e., the only difference between KATANn and KTANTANn is the key schedule part). While in the KATAN family, the 80-bit key is loaded into a register which is then repeatedly clocked, in the KTANTAN family of ciphers, the key is burnt (i.e., fixed) and the only possible "flexibility" is the choice of subkey bits. Thus, the design problem in the KTANTAN ciphers is choosing a sequence of subkeys in a secure, yet an efficient manner.

In order to minimize the hardware size, while maintaining the throughput, we treat the key as 5 words of 16 bits each. From each 16-bit word we pick the same bit (using a MUX16to1) according to the four most significant bits of the round controlling LFSR. Then, out of the five bits we choose one using the four least significant bits of the round-counting LFSR.

Formally, let $K = w_4||w_3||w_2||w_1||w_0$, where the least significant bit of w_0 is the least significant bit of K, and the most significant bit of w_4 is the most significant bit of K. We denote by T the round-counting LFSR (where T_7 is the most significant bit), then, let $a_i = \text{MUX16to1}(w_i, T_7T_6T_5T_4)$, where $\text{MUX16to1}(x, y)$ gives the yth bit of x. Then, the key bits which are used are

$$k_a = \overline{T_3} \cdot \overline{T_2} \cdot (a_0) \oplus (T_3 \vee T_2) \cdot \text{MUX4to1}(a_4a_3a_2a_1, T_1T_0),$$

$$k_b = \overline{T_3} \cdot T_2 \cdot (a_4) \oplus (T_3 \vee \overline{T_2}) \cdot \text{MUX4to1}(a_3a_2a_1a_0, \overline{T_1T_0})$$

(where $\text{MUX4to1}(x, y)$ is a MUX with 4 input bits and 1 output bit).

When considering k_a or k_b, of the 80-bit key, only one bit is used only twice, 15 are used four times, and the remaining 64 bits are used 3 times (but in total each key bit is used at least 5 times). Moreover, even if an attacker tries to pick two keys which realize the same subkey sequence for either k_a or k_b, the maximal length of such a sequence for either k_a of k_b is 35 rounds (i.e., necessarily after 35 rounds the sequences differ). We also note that due to the irregular update, during these 35 rounds, the round function is going to be different in any case.

The last issue concerning the KTANTAN key schedule is finding the most efficient way to implement it. One possible solution is to have the entire selection logic in one round. This approach requires 5 parallel MUX16to1 and our hardware implementations show that the total area consumed by the MUXes is about 180 GE. A second approach is to use one MUX16to1 and re-use it over 5 clock cycles. At a first glance, this approach may lead to a smaller circuit (while the implementation is slower). However, due to the cost of the extra control logic, this approach is not only slower, but leads to a larger circuits.

We implemented KTANTAN32 using the same *fsc0l_d_sc_tc* 0.13μm CMOS library. Our implementation requires 462 GE, of which 244 are used for the sequential logic, and 218 GE are used for the combinational logic. The simulated power consumption at 100 KHz, and throughput of 12.5 Kbps is only 146 nW. For the synthesis and the power estimation we have again used the same version of Synopsys Design Compiler.

For KTANTAN48 the implementation size of 588 GE (of which 344 are used for the sequential logic) is obtained together with the estimated power consumption of 234 nW. For the 64-bit variant, KTANTAN64, the total area is 688 GE (of which 444 are for the sequential logic) and the power consumption 292 nW. By using the clock gating as explained above, the area of 571 GE (684 GE) and the throughput of 9.4 Kb/s (8.4 Kb/s) for KATAN48 (KATAN64) is achieved.

Similar to KATAN family, we can also double (triple) a throughput for all the versions of KTANTAN family. To do that, we double (triple) the number of MUX16to1, MUX4to1, round functions f_a and f_b, and all the logic used for the feedback coefficients of the counter. Additionally, a few more gates are necessary to perform the key schedule efficiently.

6 Security Analysis

Our design philosophy was based on offering a very high level of security. To do so, we designed the ciphers with a very large security margins. For example, as a design target we have set an upper bound for the differential probability of any 128-round differential characteristic at 2^{-n} for an n-bit block size.

6.1 Differential and Linear Cryptanalysis

We have analyzed all ciphers under the assumption that the intermediate encryption values are independent. While this assumption does not necessarily hold, it simplifies the analysis and is not expected to change the results too much. Moreover, in our analysis we always take a "worst case" approach, i.e., we consider the best scenario for the attacker, which is most of the times do not happen. Hence, along with the large security margins, even if the assumption does not hold locally, it is expected that our bounds are far from being tight.

To simplify the task of identifying high probability differentials, we used computer-aided search. Our results show that depending on the used rounds, the best 42-round differential characteristic for KATAN32 has probability of 2^{-11} (it may even be lower for different set of rounds). Hence, any 126-round differential characteristic must have probability no more than $(2^{-11})^3 = 2^{-33}$. Similar results hold for linear cryptanalysis (the best 42-round linear approximation has a bias of 2^{-6}, i.e., a bias of 2^{-16} for 126-round approximation).

For KATAN48, the best 43-round differential characteristic has probability of at most 2^{-18}. Hence, any 129-round differential characteristic has probability of at most $(2^{-18})^3 = 2^{-54}$. As the probability of an active round is at least 2^{-4} this actually proves that our design criteria for 128-round differential characteristics is satisfied. The corresponding linear bias is 2^{-10} (for 43 rounds) or 2^{-28} (for 129 rounds).

Finally, repeating the analysis for KATAN64, our computer-aided search found that the best 37-round differential characteristic has probability 2^{-20}. Hence, any 111-round differential characteristic has probability of at most 2^{-60}, along with the fact that the best 18-round differential characteristic has probability of at most 2^{-5}, then the best 129-round differential characteristic has probability of no more than 2^{-65}. The linear bounds are 2^{-11} for 37 rounds and 2^{-31} for 111 rounds.

Hence, we conclude that the KATAN family is secure against differential and linear attacks. As there is no difference between the KATAN and the KTANTAN families with respect to their differential and linear behaviors, then the above is also true for the KTANTAN family.

6.2 Combined Attacks

As shown in the previous section, the probability of any differential characteristic of 128 rounds can be bounded by 2^{-n} for KATANn. Moreover, even for 64 rounds, there are no "good" characteristics. Hence, when trying to combine these together, it is unexpected to obtain good combined attacks.

For example, consider a differential-linear approximation. As noted before, the differential characteristic of 42-round KATAN32 has probability at most 2^{-11}. The bias of a 42-round KATAN32 is at most 2^{-6}. Hence, the best differential-linear property for 120 rounds is expected to have bias of at most $2 \cdot 2^{-11} \cdot (2^{-6})^2 = 2^{-22}$ (we assume a worst case assumption that allows the attacker to gain some free rounds in which the differential is truncated). Of course, an attacker may try to construct the differential-linear approximation using a different division of rounds. However, as both the probability and bias drop at least exponentially with the number of rounds, a different division is not expected to lead to better differential-linear approximations.

The same goes for the (amplified) boomerang attack. The attack (just like the differential-linear attack) treats the cipher as composed of two sub-ciphers. The probability of constructing a boomerang quartet is $\hat{p}^2\hat{q}^2$, where $\hat{p} = \sqrt{\sum_\beta \Pr{}^2[\alpha \to \beta]}$ where α is the input difference for the quartet, and β is the output difference of the characteristic in the first sub-cipher. Again, as $\hat{p}^2 \leq \max_\beta \Pr[\alpha \to \beta]$ which is bounded at 2^{-22} for 84-round KATAN32. The same goes with respect to $\hat{q}$, and thus, the probability of a boomerang quartet in 128-round KATAN32 is at most 2^{-44}.

The same rationale can be applied to KATAN48 and KATAN64, obtaining similar bounds. Specifically, the bounds for differential-linear bias is 2^{-37} (for 140 rounds) and 2^{-50} (for 160 rounds), respectively. The bounds for constructing a boomerang quartet for 128 rounds are 2^{-54} and 2^{-65}, respectively.

Another combined attack which may be considered is the impossible differential attack. This attack is based on finding a differential which has probability zero of as many rounds as possible. The most common way to construct such a differential is in a miss-in-the-middle manner, which is based on finding two (truncated) differentials with probability 1 which cannot co-exist. Due to the quick diffusion, changing even one bit would necessarily affects all bits after at most 42 rounds (37 for KATAN48 and 38 for KATAN64), and thus, there is no impossible differential of more than 168 rounds (after 42 rounds, change of any bit may affect all bits, and thus, after 84 rounds, each differential may have any output difference).

Hence, we conclude that the KATAN family (as well as the KTANTAN family) of block ciphers is secure against combined attacks.

6.3 Slide and Related-Key Attacks

As mentioned before, the way KATAN and KTANTAN were designed to foil self-similarity attacks by using two types of rounds which are interleaved in a non-repeating manner.

First, consider the slide attack, which is based on finding two messages such that they share most of the encryption process (which are some rounds apart). Given the fact that there is a difference between the deployed round functions, this is possible only for a very small number of rounds. Even if we allow these relations to be probabilistic in nature (i.e., assume that the bit of the intermediate value is set to 0 thus preventing the change in the function to change the similarity between the states). For example, when considering KATAN32, there is no slide property with probability 2^{-32} starting from the first round of the cipher. The first round from which such a property can be constructed is round 19. If an attacker achieves the same intermediate encryption value after round 19 and round 118, he may find a "slid" pair which maintains the equality with probability 2^{-31} until the end of the cipher (i.e., the output of the second encryption process will be the same as the intermediate encryption value of the first encryption at round 155). This proves that there are no good slid properties in the cipher family (we note that this probability is based on the assumption that the subkeys are the same, which is not the case, unless the key is the all zeros key). When it comes to KATAN48 or KATAN64, this probability is even lower (as there are more bits which need to be equal to zero), i.e., 2^{-62} and 2^{-93}, respectively, rendering slide attacks futile against the KATAN and the KTANTAN families (these values are actually an upper bound as they assume that all the subkeys are the same).

Now consider a related-key attack. In the related-key settings, the attacker searches for two intermediate encryption values as well as keys which develop in the same manner for as many rounds as possible. As noted before, there are no "good" relations over different rounds, which means that the two intermediate encryption values have to be in the same round. However, by changing even one singly bit of the key causes a difference after at most 80 rounds of similar encryption process. Hence, no related-key plaintext pairs (or intermediate encryption values) exist for more than 80 rounds (similarity in 80 rounds would force the key and the intermediate encryption value to be the same). As this is independent of the actual key schedule algorithm, it is easy to see that both the KATAN and the KTANTAN families are secure against this attack.

The only attack in this category which remains is related-key differential attack. This is the only attack where there is a difference between the two families of ciphers according to their key schedule algorithm. We first consider the case of the KATAN family. The key schedule of the KATAN family expands linearly the 80-bit key into 508 subkey bits (each is used once in KATAN32, twice in KATAN48, and thrice in KATAN64). We note that the probability of the differential is reduced any time a difference enters one of the nonlinear functions (i.e., the AND operation). Thus, it is evident that good related-key differentials have as little active bits as possible. Moreover, we can relate the number of active bits throughout the encryption process to the issue of active bits in the key schedule. Each active bit of the subkey (i.e., a subkey bit with a difference) either causes a difference in the internal state (which in turn incurs probability and activation of more bits), or is being canceled by previous differences. We note that each active subkey bit which is not canceled, necessarily induces probability "penalty" of 2^{-2} in KATAN32, 2^{-4} in KATAN48, and 2^{-6} in KATAN64. Moreover, due to the way the cipher works, each active bit can "cancel" at most four other active subkey bits.[4] Hence, if the weight of the expanded subkey difference is more than 80, then it is assured that the probability of any related-key differential of KATAN32 is at most 2^{-32} (this follows the fact that each active bit in the intermediate encryption value may cancel up to four subkey bit differences injected, and we shall assume a worst case assumption that the positions align

[4] We note that five or six can be canceled, but in this case, the probability penalty of an active bit is increased by more than the "gain" offered by using this active bit more times

"correctly). For KATAN48, due to the increased penalty, it is sufficient that the minimal weight of the expanded subkey difference is more than 60, and for KATAN64 the minimal weight needs to be at least 54. We were analyzing the minimal weight using the MAGMA software package, and the current bounds are between 72 and 84. Hence, we conclude that the KATAN family of block ciphers is expected to be resistant to related-key differential attacks.

For the KTANTAN family, due to the fixed key, the concept of related-key attacks is of theoretical interest. Still, we can follow a more detailed analysis using the same ideas as we used for regular differential searches. While the search space is huge, our current results show that there is no related-key differential characteristic for more than 150 rounds of KTANTAN32 with probability greater than 2^{-32}. Similar results are expected to hold for KTANTAN48 and KTANTAN64.

6.4 Cube Attacks and Algebraic Attacks

Given the low algebraic degree of the combining function, it may look as if KATAN and KTANTAN are susceptible to algebraic attacks or cube attack [7]. However, when considering the degree of the expressions involving the plaintext, one can see that after 32 rounds (for KATAN32) the degree of each internal state bit is at least 2, which means that after 160 rounds, the degree of each internal state bit can reach 32. For KATAN48, the degree is at least 2 after 24 rounds, (or about 48 after 144 rounds), and for KATAN64 it is 2 after 22 rounds and can reach 64 after 132 rounds). Hence, as the degree can reach to the maximal possible value (and there are some more rounds to spare), it is expected that the KATAN and KTANTAN families are secure against algebraic attacks.

Another possible attack is the cube attack, which was successful against reduced-round variants of Trivium (with less initialization rounds than in the Trivium). We note that in trivium the internal state is clocked four full cycles (i.e., each bit traverse all the registers exactly four times). In KATAN32, most bits traverse the registers eight times (where a few does so only seven times), going through more nonlinear combiners (each Trivium round uses only one AND operation per updated bit), and thus is expected to be more secure against this attack than Trivium. The same is also true for KATAN48 (where about half of the bits traverse the registers 10 times, and the other bits do so 11 times) and KATAN64 (where most of the bits traverse the registers 12 times, and a few do that only 11 times).

7 Summary

In this paper we have presented two families of hardware efficient encryption algorithms. The family of cipher is suitable for low-end devices, and even offer algorithmic security level of 80 bits in cases where the key is burnt into the device (of course, the implementation has to be protected as well). As the proposal have a simple structure and use very basic components, it appears that common techniques to protect the implementation should be easily adopted.

References

1. Eli Biham, *New Types of Cryptanalytic Attacks Using Related Keys*, Journal of Cryptology, Vol. 7, No. 4, pp. 229–246, Springer, 1994.
2. Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer, 1993.

3. Alex Biryukov, David Wagner, *Slide Attacks*, Proceedings of Fast Software Encryption 6, Lecture Notes in Computer Science 1636, pp. 245–259, Springer, 1999.
4. Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, Charlotte H. Vikkelsoe, *PRESENT: An Ultra-Lightweight Block Cipher*, Proceedings of Cryptographic Hardware and Embedded Systems (CHES) 2007, Lecture Notes in Computer Science 4727, pp. 450–466, Springer, 2007.
5. Nicolas T. Courtois, Gregory V. Bard, David Wagner, *Algebraic and Slide Attacks on KeeLoq*, Proceedings of Fast Software Encryption 15, to appear in Lecture Notes in Computer Science, Springer.
6. Christophe De Canniére, Bart Preneel, *Trivium Specifications*, eSTREAM submission, available online at *http://www.ecrypt.eu.org/stream/triviump3.html.*
7. Itai Dinur, Adi Shamir, *Cube Attacks on Tweakable Black Box Polynomials*, IACR ePrint report 2008/385, accepted to EUROCRYPT 2009.
8. Martin Feldhofer, Johannes Wolfkerstorfer, Vincent Rijmen *AES implementation on a grain of sand*, IEE Proceedings of Information Security, , Vol. 152, No. 1, pp. 13–20, IEE, 2005.
9. Tim Good, Mohammed Benaissa, *Hardware results for selected stream cipher candidates*, preproceedings of SASC 2007, pp. 191–204, 2007.
10. Panu Hämäläinen, Timo Alho, Marko Hännikäinen, Timo D. Hämäläinen, *Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core*, Ninth Euromicro Conference on Digital System Design: Architectures, Methods and Tools, IEEE Computer Society, 2006.
11. Martin Hell, Thomas Johansson, Willi Meier, *Grain — A Stream Cipher for Constrained Environments*, eSTREAM submission, available online at *http://www.ecrypt.eu.org/stream/p3ciphers/grain/Grain_p3.pdf.*
12. Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bon-Seok Koo, Changhoon Lee, Donghoon Chang, Jesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim, Seongtaek Chee, *HIGHT: A New Block Cipher Suitable for Low-Resource Device*, Proceedings of Cryptographic Hardware and Embedded Systems (CHES) 2006, Lecture Notes in Computer Science 4249, pp. 46–59, Springer, 2006.
13. Sebastiaan Indesteege, Nathan Keller, Orr Dunkelman, Eli Biham, Bart Preneel, *A Practical Attack on KeeLoq*, Advances in Cryptography, EUROCRYPT 2008, to appear in Lecture Notes in Computer Science, Springer. *Private communications.*
14. Suzan K. Langford, Martin E. Hellman, *Differential-Linear Cryptanalysis*, Advances in Cryptology, Proceedings of CRYPTO '94, Lecture Notes in Computer Science 839, pp. 17–25, Springer, 1994.
15. Chae Hoon Lim, Tymur Korkishko, *mCrypton — A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors*, Proceedings of Workshop on Information Security Applications (WISA) 2005, Lecture Notes in Computer Science 3786, pp. 243–258, Springer, 2005.
16. Nele Mentens, Jan Genoe, Bart Preneel, Ingrid Verbauwhede, *A low-cost implementation of Trivium*, preproceedings of SASC 2008, pp. 197–204, 2008.
17. Microchip Technology Inc. *KeeLoq*® *Authentication Products*, `http://www.microchip.com/keeloq/`
18. Mitsuru Matsui, *Linear Cryptanalysis Method for DES Cipher*, Advances in Cryptology, Proceedings of EUROCRYPT '93, Lecture Notes in Computer Science 765, pp. 386–397, Springer, 1994.
19. Axel Poschmann, Gregor Leander, Kai Schramm, Christof Paar, *New Light-Weight DES Variants Suited for RFID Applications*, Proceedings of Fast Software Encryption 2007, Lecture Notes in Computer Science 4593, pp. 196–210, Springer, 2007.
20. Carsten Rolfes, Axel Poschmann, Gregor Leander, Christof Paar, *Ultra-Lightweight Implementations for Smart Devices — Security for 1000 Gate Equivalents*, Proceedings of Smart Card Research and Advanced Applications (CARDIS) 2008, Lecture Notes in Computer Science 5189, pp. 89–103, Springer, 2008.

Rounds	0–9	10–19	20–29	30–39	40–49	50–59
Irregular	1111111000	1101010101	1110110011	0010100100	0100011000	1111000010
Rounds	60–69	70–79	80–89	90–99	100–109	110–119
Irregular	0001010000	0111110011	1111010100	0101010011	0000110011	1011111011
Rounds	120–129	130–139	140–149	150–159	160–169	170–179
Irregular	1010010101	1010011100	1101100010	1110110111	1001011011	0101110010
Rounds	180–189	190–199	200–209	210–219	220–229	230–239
Irregular	0100110100	0111000100	1111010000	1110101100	0001011001	0000001101
Rounds	240–249	250–253				
Irregular	1100000001	0010				

Table 3. The sequence of the irregular updates. 1 means that the irregular update rule is used in this round, while 0 means that this is not the case.

A The Irregular Update Sequence

B Implementation Trade-Offs

Table 4. Area-Throughput Trade-Offs.

Cipher	Block (bits)	Key (bits)	Size (GE)	Gates per Memory Bit	Throughput* (Kb/s)	Logic Process
KATAN32	32	80	802	6.25	12.5	0.13 μm
KATAN32	32	80	846	6.25	25	0.13 μm
KATAN32	32	80	898	6.25	37.5	0.13 μm
KATAN48†	48	80	916	6.25	9.4	0.13 μm
KATAN48	48	80	927	6.25	18.8	0.13 μm
KATAN48	48	80	1002	6.25	37.6	0.13 μm
KATAN48	48	80	1080	6.25	56.4	0.13 μm
KATAN64†	64	80	1027	6.25	8.4	0.13 μm
KATAN64	64	80	1054	6.25	25.1	0.13 μm
KATAN64	64	80	1189	6.25	50.2	0.13 μm
KATAN64	64	80	1269	6.25	75.3	0.13 μm
KTANTAN32	32	80	462	6.25	12.5	0.13 μm
KTANTAN32	32	80	673	6.25	25	0.13 μm
KTANTAN32	32	80	890	6.25	37.5	0.13 μm
KTANTAN48†	48	80	571	6.25	9.4	0.13 μm
KTANTAN48	48	80	588	6.25	18.8	0.13 μm
KTANTAN48	48	80	827	6.25	37.6	0.13 μm
KTANTAN48	48	80	1070	6.25	56.4	0.13 μm
KTANTAN64†	64	80	684	6.25	8.4	0.13 μm
KTANTAN64	64	80	688	6.25	25.1	0.13 μm
KTANTAN64	64	80	927	6.25	50.2	0.13 μm
KTANTAN64	64	80	1168	6.25	75.3	0.13 μm

* — A throughput is estimated for frequency of 100 KHz.
† — Using clock gating.