A comparison of block ciphers SIMON, SPECK, and KATAN

Andreas Bossert[1], Steven Cooper[1], and Alexander Wiesmaier[1,2,3]

[1] TU Darmstadt
[2] AGT International
[3] Hochschule Darmstadt

Abstract. In this paper we present 3 block cipher algorithms `Simon`, `Speck` and `KATAN`. Each of them gets a short introduction of their functions and recommended field of use. We also compare these 3 block ciphers with each other and with the state of the art algorithm the Advanced Encryption Standard (AES) to see how efficient and fast they are to be able to conclude what algorithm is the best for which specific application.

Keywords: Internet of things (IoT); lightweight block ciphers; SIMON; SPECK; KATAN

1 Introduction

In modern IT `The Internet of Things(IoT)` is one of the most recent topics. More and more devices get functions to go online interconnect with each other and send and receive data. And it is very important to be sure that these connections are secure but also efficient. The state of the art block cipher AES cannot be used for these low-end devices such as RFID tags or sensor networks because they are often very small, have less computing power or have to be very power saving. So we have very constrained environments. Our 3 block ciphers (SIMON,SPECK and KATAN) had been developed to fulfill these constraints. KATAN and SIMON have been optimized for performance on hardware devices and SPECK for performance in software. At the comparison section we will see how good they fulfill these goals. The focus is on the comparison of SIMON, SPECK and KATAN and the compare with AES.

We organize this paper as follows: In Section 2 we have a short overview of further related works which also concerned about our 3 algorithms. Section 3, 4 and 5 presents SIMON, SPECK and KATAN. How they work, the different variants and possible attacks against them. In Section 6 we have the comparison of the algorithms and our conclusion.

2 Related Work

Our work focuses on three Block ciphers that are compared to each other and set in context to AES and known attacks on them. Papers which handle similar content are [1], [2], [3] and [4].

In the area of SIMON and SPECK there exist two similar papers "The SIMON and SPECK Families of Lightweight Block Ciphers" [1] and "SIMON and SPECK: Block Ciphers for the Internet of Things" [2]. The papers explains how the algorithm works, it contains many performance comparisons on constrained platforms and it describes many security aspects. A few of the comparisons were used in this paper too.

The only paper which compares `KATAN` with other lightweight block ciphers is the paper from the designers of `KATAN` [3]. It explains what are the difference between KATAN and several existing block ciphers and compares them. The paper [4] handles among other about different attacks on KATAN and compares them with each other. Furthermore does it explain how KATAN is designed and explains a new algebraic attack which is better than any other known algebraic attacks. We use these results in our own comparison of the block ciphers and the attacks on them and create a paper where it is all in one.

3 Simon and Speck

Simon and Speck is a family of lightweight block ciphers publicly released by the National Security Agency (NSA) in June 2013. [1] Simon and Speck comes with ten distinct block ciphers with differing block and key sizes. The most existing block ciphers were designed to perform well on a single platform and were not meant to provide high performance across a range of devices. The aim of Simon and Speck was to fill the need for secure, flexible, and analyzable lightweight block ciphers. Each offers excellent performance on hardware and software platforms, is flexible enough to admit a variety of implementations on a given platform, and is amenable to analysis using existing techniques. Both perform very well across the full spectrum of lightweight applications, but Simon has been optimized for performance in hardware implementations, while its sister algorithm, Speck, has been optimized for software implementations. The reason why the algorithms work so well on each platform is that both are very simple constructed. So it is very easy to find efficient implementations. For algorithms such as AES it required longer time of research to find near-optimal implementations.

The Simon block cipher with an n-bit word (and hence a 2n-bit block) is denoted Simon2n, where n is required to be 16, 24, 32, 48, or 64. Simon2n with an m-word (mn-bit) key will be referred to as Simon2n/mn. For example, Simon64/128 refers to the version of Simon acting on 64-bit plaintext blocks and using a 128-bit key. The notation for the different variants of Speck is entirely analogous to that used for Simon.

3.1 Simon round function

The Simon2n encryption maps make use of the following operations on n-bit words:

- bitwise XOR, $\oplus$,
- bitwise AND, &, and
- left circular shift, S^j , by j bits.

The round functions for Simon 2n take as input an n-bit round key k, together with two n-bit intermediate ciphertext words. The round function is the 2-stage Feistel map

$$R_k(x, y) = (y \oplus f(x) \oplus k, x),$$

where $f(x) = (Sx \& S^8 x) \oplus S^2 x$ and k is the round key. The inverse of the round function, used for decryption, is

$$R_k^{-1}(x, y) = (y, x \oplus f(y) \oplus k).$$

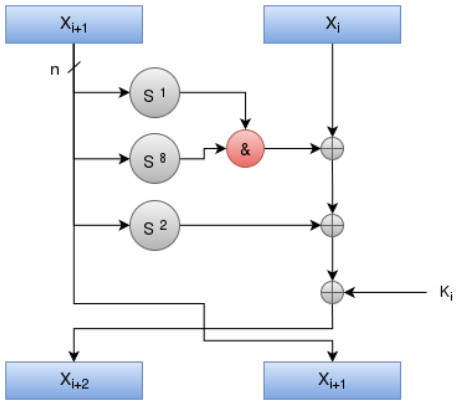

Fig. 1. The Simon Round Function. Derived from [1]

Figure 1 shows the effect of the round function R_{k_i} on the two words of sub cipher (x_{i+1}, x_i) at the i^{th} step of this process.

The round functions are composed some number of times which depends on the block and key size. Parameters for all versions of Simon are specified in Table 1.

Table 1. SIMON parameters. Derived from [1]

block size 2n	key size mn	word size n	key words m	const seq	rounds T
32	64	16	4	z_0	32
48	72	24	3	z_0	36
	96		4	z_1	36
64	96	32	3	z_2	42
	128		4	z_3	44
96	96	48	2	z_2	52
	144		3	z_3	54
128	128	64	2	z_2	68
	192		3	z_3	69
	256		4	z_4	72

3.2 Simon key schedules

The key schedule is needed to turn a key into a sequence of round keys. The Simon key schedules employ a sequence of 1-bit round constants specifically for the purpose of eliminating slide properties and circular shift symmetries.

The designers provide some cryptographic separation between different versions of Simon having the same block size by defining five such sequences: $z_0, ..., z_4$. Each of these sequences is defined in terms of one of the following period 31 sequences:

$$\begin{aligned} u = u_0u_1u_2... &= 11111010001001010110000111001 10..., \\ v = v_0v_1v_2... &= 10001110111110010011000010110 10..., \\ w = w_0w_1w_2... &= 10000100101100111110001101110 10.... \end{aligned}$$

The first two sequences are simply $z_0 = u$ and $z_1 = v$. The other three, z_2, z_3, and z_4, have period 62 and are formed by computing the bitwise XOR of the period 2 sequence $t = t_0t_1t_2... = 01010101...$ with u, v, and w, respectively:

$$\begin{aligned} z_2 = (z_2)_0(z_2)_1(z_2)_2... &= 10101111011100000011010010011 00 \\ &\quad 01010000100011111100101101100 11..., \\ z_3 = (z_3)_0(z_3)_1(z_3)_2... &= 11011011101011000110010111100 00 \\ &\quad 00100100010100111001101000011 11..., \\ z_4 = (z_4)_0(z_4)_1(z_4)_2... &= 11010001111001101011011000100 00 \\ &\quad 00101110000110010100100111011 11..., \end{aligned} \tag{1}$$

where $(z_i)_j$ is the j^{th} bit of z_i.
The sequences u, v, and w can be generated as follows: Define 5×5 matrices U, V, and W over GF(2) by

$$U = \begin{bmatrix} 0&1&0&0&0 \\ 0&0&1&0&0 \\ 1&0&0&1&0 \\ 0&0&0&0&1 \\ 1&0&0&0&1 \end{bmatrix}, V = \begin{bmatrix} 0&1&1&0&0 \\ 0&0&1&0&0 \\ 1&0&0&1&0 \\ 0&0&0&0&1 \\ 1&0&0&0&0 \end{bmatrix}, W = \begin{bmatrix} 0&1&0&0&0 \\ 0&0&1&0&0 \\ 1&0&0&1&0 \\ 0&0&0&0&1 \\ 1&0&0&0&0 \end{bmatrix}$$

The i_{th} element of each sequence is then obtained by initializing a 5-bit linear feedback shift register to 00001, stepping i times using the corresponding matrix, and extracting the right-hand bit. Thus $(u)_i = (0,0,0,0,1)U^i(0,0,0,0,1)^t$.

Let $c = 2^n - 4 = 0xff\cdots fc$. For Simon2n with m key words $(k_{m-1}, ..., k_1, k_0)$ and constant sequence z_j, round keys are generated by

$$k_{i+m} = \begin{cases} c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})S^{-3}k_{i+1}, & \text{if } m = 2 \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})S^{-3}k_{i+2}, & \text{if } m = 3 \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})(S^{-3}k_{i+3} \oplus k_{i+1}, & \text{if } m = 4 \end{cases}$$

for $0 \leq i < T - m$. In Figure 2 is the key schedules represented and which version-dependent choice of constant sequence z_j have to used is shown in Table 1. Note that yourself choose the first m key words which will used as the first m round keys. They are loaded into the shift registers with k_0 on the right and k_{m-1} on the left. Only the next ones will be generated with key schedule.

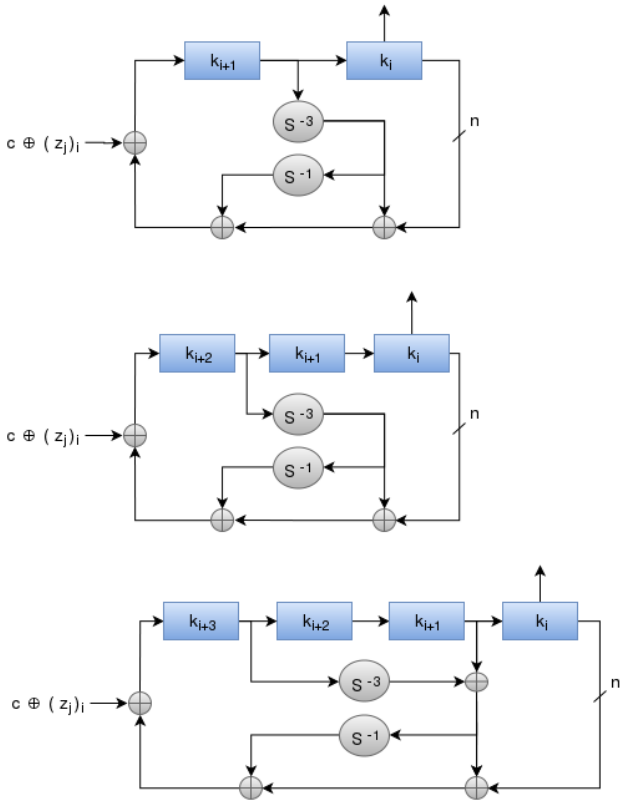

Fig. 2. The Simon two, three, and four-word key expansion. Derived from [1]

3.3 Speck round function

The Speck2n encryption maps make use of the following operations on n-bit words:

- bitwise XOR, $\oplus$,
- addition modulo 2^n, $+$,
- left and right circular shifts, S^j and $S^{-}j$, respectively, by j bits.

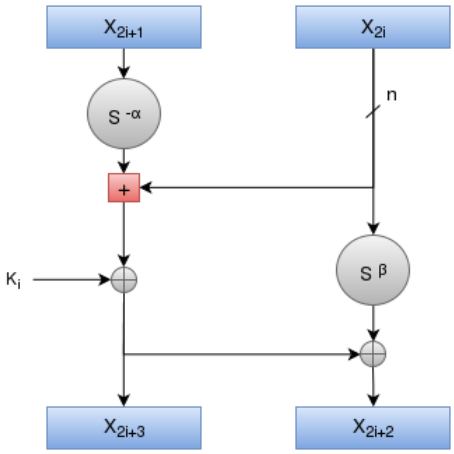

Fig. 3. Speck round function; (x_{2i+1} , x_{2i}) denotes the sub-cipher after i steps of encryption. Derived from [1]

For $k \in GF(2)^n$, the key-dependent Speck2n round function is the map R_k: GF$(2)^n\times$ GF$(2)^n \rightarrow$ GF$(2)^n\times$ GF$(2)^n$ defined by

$$R_k(x, y) = ((S^{-\alpha}x + y) \oplus k, S^{\beta}y \oplus (S^{-\alpha}x + y) \oplus k),$$

with rotation amounts $\alpha = 7$ and $\beta = 2$ if $n = 16$ (block size $= 32$) and $\alpha = 8$ and $\beta = 3$ otherwise.

The inverse of the round function, necessary for decryption, uses modular subtraction instead of modular addition, and is given by

$$R_k^{-1}(x, y) = (S^{\alpha}((x \oplus k) - S^{-\beta}(x \oplus y)), S^{-\beta}(x \oplus y)),$$

Parameters for all versions of Speck are specified in Table 2.

The Speck key schedules take a key and from it generate a sequence of T key words $k_0, ..., k_{T-1}$, where T is the number of rounds. The effect of the single

Table 2. Speck parameters. Derived from [1]

block size 2n	key size mn	word size n	key words m	rot α	rot β	rounds T
32	64	16	4	7	2	22
48	72	24	3	8	3	22
	96		4			23
64	96	32	3	8	3	26
	128		4			27
96	96	48	2	8	3	28
	144		3			29
128	128	64	2	8	3	32
	192		3			33
	256		4			34

round function R_{k_i} is shown in Figure 3. Encryption is then the composition $R_{k_{T-1}} \circ ... \circ R_{k_1} \circ R_{k_0}$, read from right to left.
Note that Speck can be realized as the composition of two Feistel-like maps with respect to two different types of addition, namely,

$$(x, y) \mapsto (y, (S^{-\alpha}x + y) \oplus k) and (x, y) \mapsto (y, S^{\beta}x \oplus y).$$

This decomposition is pictured in Figure 4.

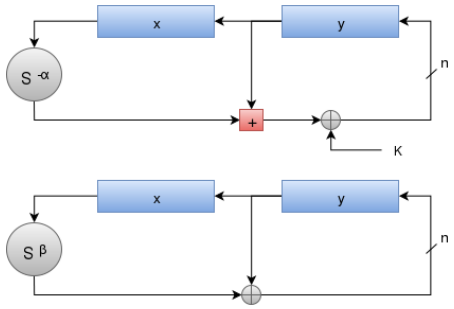

Fig. 4. Speck round function decomposed into Feistel-like steps. Derived from [1]

3.4 Speck key schedules

The Speck key schedules use the own round function to generate round keys k_i. This is useful cause we don't need to implement a new method. Let K be a key for a Speck2n block cipher. We can write $K = (l_{m-2}, ..., l_0, k_0)$, where $l_i, k_0 \in GF(2)^n$, for a value of m in 2, 3, 4. Sequences k_i and l_i are defined by

$$l_{i+m-1} = (k_i + S^{-\alpha}l_i) \oplus i$$

and

$$k_{i+1} = S^{\beta} k_i \oplus l_{i+m-1}.$$

The value k_i is the i^{th} round key, for $0 \leq i < T$. See Figure 5.

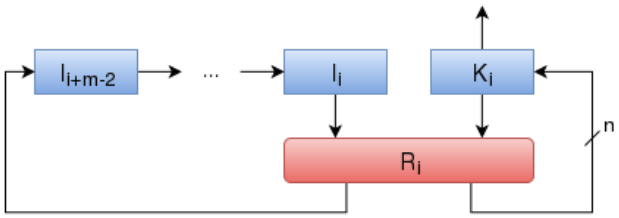

Fig. 5. Speck key expansion, where R_i is the Speck round function with i acting as round key. Derived from [1]

3.5 Security Analysis

Simon and Speck attacks was studied in many articles since its publication in 2013. The most published attacks on Simon and Speck are of the reduced-round variety. The goal of this sort of analysis is to determine the maximal number of rounds that would be susceptible to a theoretical attack (i.e., anything better than an exhaustive key search). A measure of security is the number of rounds that have been attacked, as a percentage of the total. So far no published attack makes it more than about 74% of the way through any version of Simon or Speck. The best attacked rounds for all versions of Simon was with the improved linear hull and differential attacks with dynamic key-guessing technique [5], [6] . The best attacked rounds in Speck is with differential cryptanalysis and improved differential cryptanalysis [7], [8]. The results are shown in table 3.

The content of the table 3 is simple: there are no attacks on any member of the Simon or Speck families, and each block cipher maintains a healthy security margin.

Table 3. Security of Simon and Speck derived from "Simon and Speck: Block Ciphers for the Internet of Things" [2] Table 1

size	alg	rounds		ref
		total	attacked	
32/64	Simon	32	23 (72%)	[5]
	Speck	22	14 (64%)	[7]
48/72	Simon	36	24 (67%)	[5]
	Speck	22	14 (64%)	[7]
48/96	Simon	36	25 (69%)	[5]
	Speck	23	15 (65%)	[7]
64/96	Simon	42	30 (71%)	[5]
	Speck	26	18 (69%)	[7]
64/128	Simon	44	31 (70%)	[5]
	Speck	27	19 (70%)	[7]
96/96	Simon	52	37 (71%)	[6],[5]
	Speck	28	16 (57%)	[7]
96/144	Simon	54	38 (70%)	[5]
	Speck	29	17 (59%)	[7]
128/128	Simon	68	49 (72%)	[6],[5]
	Speck	32	17 (53%)	[7]
128/192	Simon	69	51 (74%)	[5]
	Speck	33	18 (55%)	[7],[8]
128/256	Simon	72	53 (74%)	[5]
	Speck	34	19 (56%)	

4 KATAN

KATAN/KTANTAN is a family of hardware oriented block ciphers designed in 2009 by Chrstophe de Canniere, Orr Dunkelman, and Miroslav Knezevic [3]. In summary the family consists of six block ciphers. They are divided into two sets of three KATAN block ciphers with 32, 48 or 64-bit block size and three KTANTAN block ciphers with the same block size. They share the same 80-bit key size and security level. The difference between KATAN and KTANTAN is that at KTANTAN the key is burnt into the device and cannot be changed. Therefor KTANTAN are very small block ciphers and more compact than KATAN and can only be used in cases where the device is initialized with one key.

- KATAN32 has 802 GE and an encryption speed of 12.5 KBit/sec.
- KATAN48 has 927 GE and an encryption speed of 18.8 KBit/sec.
- KATAN64 has 1054 GE and an encryption speed of 25.1 KBit/sec.
- KTANTAN32 has 462 GE and an encryption speed of 12.5 KBit/sec.
- KTANTAN48, which is the recommend for RFID tags has 588 GE and an encryption speed of 18.8 KBit/sec.
- KTANTAN64 has 688 GE and an encryption speed of 25.1 KBit/sec.

A comparison with some other ciphers is shown in Table 4.

Table 4. Comparison of Ciphers Designed for Low-End Environments (optimized for size). Derived from[3].

Cipher	Block (bits)	Key (bits)	Size (GE)	Gates per Memory Bit	Throughput[1] (Kb/s)	Logic Process
AES-128	128	128	3400	7.97	12.4	0.35μ
AES-128	128	128	3100	5.8	0.08	0.13μ
HIGHT	64	128	3048	N/A	188.25	0.25μ
mCrypton	64	64	2420	5	492.3	0.13μ
DES	64	56	2309[2]	12.19	44.4	0.18μ
DESL	64	56	1848[2]	12.19	44.4	0.18μ
PRESENT-80	64	80	1570	6	200	0.18μ
PRESENT-80	64	80	1000	N/A	11.4	0.35μ
Grain	1	80	1294	7.25	100	0.13μ
Trvivium	1	80	749	2[3]	100[4]	0.35μ
KATAN32	32	80	802	6.25	12.5	0.13μ
KATAN48	48	80	927	6.25	18.8	0.13μ
KATAN64	64	80	1054	6.25	25.1	0.13μ
KTANTAN32	32	80	462	6.25	12.5	0.13μ
KTANTAN48	48	80	588	6.25	18.8	0.13μ
KTANTAN64	64	80	688	6.25	25.1	0.13μ

[1] —A throughput is estimated for frequency of 100 KHz.
[2] —Fully serialized implementation (the rest are only synthesized).
[3] —This is a full-custom design using C2MOS dynamic logic.
[4] —This throughput is projected, as the chip requires higher frequencies.

The specific design goals from the developers were as follows:[3]

- 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 280 exists on the entire cipher.
- High enough algebraic degree for the equation describing half the cipher to thwart any algebraic attack.

Also they rank the possible design targets as follows:[3]

- 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 footprint is small).
- Decrease the power consumption of the implementation.

We concentrate on `KATAN` in this paper so `KTANTAN` is not examined in more detail.

4.1 Round function and key schedule

We have three variants of the `KATAN` ciphers. `KATAN32`, `KATAN48` and `KATAN64`. The main difference between them is the block size and that `KATAN48` executes the nonlinear function twice and `KATAN64` three times with the same round key per round. For example, we use `KATAN32` to describe the key schedule. The plaintext (bit 0-31) is used to generate the ciphertext. For that it is loaded into two registers L_1(bit 19-31) and L_2(bit 0-18) and L_1 and L_2 are shifted to the left (bit i is shifted to position i+i) each round. After that both registers get updated each round with the following nonlinear functions f_a and f_b for 254 rounds.

$$f_a(L_1) = L_1[x_1] \oplus L_1[x_2] \oplus (L_1[x_3] * L_1[x_4]) \oplus (L_1[x_5] * IR) \oplus k_a$$
$$f_b(L_2) = L_2[y_1] \oplus L_2[y_2] \oplus (L_2[y_3] * L_2[y_4]) \oplus (L_2[y_5] * L_2[y_6]) \oplus k_b$$

IR is an irregular update rule (which is only used if IR = 1) shown in Table 6, k_a and k_b are two subkey bits. The bits for x_1 and y_1 for each variant are shown in Table 5[3]. Afther 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 for all variants of the `KATAN` family accepts a 80-bit key K with the secret key K_0 - K_{79} and the following mapping:

$$k_i = \begin{cases} K_i & \text{for } i = 0...79 \\ k_{i-80} \oplus k_{i-61} = \oplus k_{i-50} \oplus k_{i-13} & \text{Otherwise} \end{cases} \quad (2)$$

The values of k_a and k_b for a round `i` are k_{2i} and k_{2+i}. And for that $k_a \mid\mid k_b = k_{2i} \mid\mid k_{2+i}$. Figure 6[3] shows a round of the KATAN family.

Table 5. Parameters of the KATAN family. Derived from [3].

Cipher	$\|L_1\|$	$\|L_2\|$	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	23	33	21	14	9

Table 6. Sequence of the irregular updates. 1 IR is used, 0 IR is not used. Derived from [3].

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				

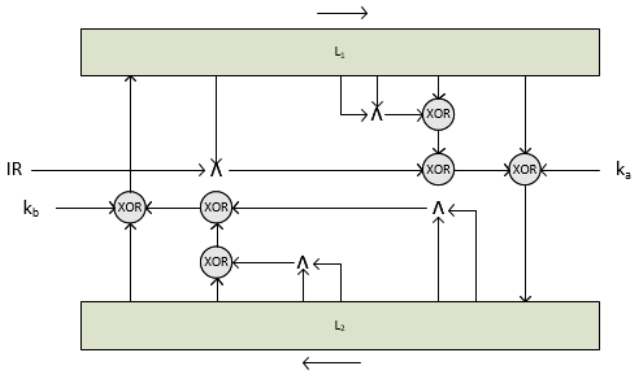

Fig. 6. Outline of a round of the KATAN family. Derived from [3].

4.2 Security Analysis

The first two mentioned design goals ensure that no differential-linear attack or a boomerang attack exist for the entire cipher. Only one successful attack is known at the moment. The attack on KTANTAN32 was presented by Andrey Bogdanov and Christian Reichberger at Selected Areas in Cryptography 2010 [9]. The meet-in-the-middle attack can find the key with a time complexity of 2^{79}. The other variants of the KATAN family are not affected by this attack and still secure. Mainly differential, meet-in-the-middle, algebraic and side channel attacks have been executed on KATAN. First we look at 2 differential attacks.

The authors from [10] used a known chosen plaintext scenario with multiple instances with the same key to attack KATAN. They get 16 differentials for 95 rounds which makes it possible to break 115 rounds of KATAN with a time and data complexity of 2^{32}.

In [11] multiple KATAN instances with a difference in plaintext and key. The attack breaks 120 rounds of KATAN with a time and data complexity of 2^{31}.

The third attack is a meet-in-the-middle-attack from [12]. They break 153 rounds from KATAN with a time and data complexity of $2^{78.5}$ and a memory complexity of 2^{76}. So this is a more theoretical attack and not practicable with the current technologies.

Another meet-in-the-middle-attack is from [13]. 174 rounds are broken with a time complexity of $2^{78.5}$, a memory complexity of $2^{26.6}$ and a data complexity of $2^{27.6}$.

An algebraic attack is from [14]. The attack breaks 79 rounds with a time complexity of $2^{76.5}$ and a data complexity of 2^5.

A little better is the algebraic attack from [4]. The attack breaks 80 rounds with a time complexity of 2^{72} and a data complexity of 2^7.

In Table 7 a comparison of the attacks is shown.

Table 7. Comparison of the attacks

attack	method	rounds	time	memory	data
[10]	differential	115	2^{32}	—	2^{32}
[11]	differential	120	2^{31}	—	2^{31}
[12]	meet-in-the-middle	153	$2^{78.5}$	2^{76}	$2^{78.5}$
[13]	meet-in-the-middle	174	$2^{78.5}$	$2^{26.6}$	$2^{27.6}$
[14]	algebraic	79	$2^{76.5}$	—	2^5
[4]	algebraic	80	2^{72}	—	2^7

5 Comparison and Conclusion

5.1 Comparison

Table 8 shows a comparison of Simon, Speck, AES, KATAN and a few others. As we can see Simon is the best in hardware size. It requires the least GE in all different block/key size versions but KATAN has the better efficiency (Throughput/GE). In software we can see that Speck has the best throughput and has the lowest memory usage.

5.2 Conclusion

Our conclusion is that Simon and Speck are over all our recommended block ciphers. The advantage from Simon and Speck is the simplicity and flexibility. These two properties make it possible to implement the algorithms in different ways. The algorithms can be very small to run on FPGA, microcontroller, and microprocessor implementations, but can also achieve very high throughput on all of these platforms.
If you don't trust Simon and Speck because of the NSA background and the aforementioned or other reasons, KATAN is a very good alternative for hardware implementations and performs very near to Simon. KATAN can be very efficiency, so if the area space is not very limited and a high throughput is needed KATAN is the right alternative choice. Also you can choose between 3 variants of KATAN depending on the needed security level.

Table 8. Performance comparisons. Hardware refers to an ASIC implementation, and software to an implementation on an 8-bit micro- controller; clock speeds are 100 kHz (hardware) and 16 MHz (software). The best performance for a given size is indicated in red, the second best in blue. Derived from "Simon and Speck: Block Ciphers for the Internet of Things" [2] table 1.1 and [3]

		hardware			software		
size	name	area (GE)	throughput (kbps)	efficiency (kbps/GE)	flash (bytes)	SRAM (bytes)	throughput (kbps)
32/80	KATAN32	802	12.5	16	-	-	-
	KATAN32	846	25	30	-	-	-
	KATAN32	898	37.5	37	-	-	-
48/80	KATAN48	927	18.8	20	-	-	-
	KATAN48	1002	37.6	38	-	-	-
	KATAN48	1080	56.4	61	-	-	-
48/96	SIMON	763	15.0	20	196	0	589
	SPECK	884	12.0	14	134	0	943
	EPCBC	1008	12.1	12	[365]	0	[93]
64/80	TWINE	1011	16.2	16	1304	414	472
	PRESENT	1030	12.4	12	[487]	0	96
	PICCOLO	1043	14.8	14	-	-	-
	KATAN64	1054	25.1	24	272	18	14
	KATAN64	1189	50.2	42	-	-	-
	KATAN64	1269	75.3	59	-	-	-
	KLEIN	1478	23.6	16	766	18	168
64/96	SIMON	838	17.8	21	274	0	540
	SPECK	984	14.5	15	182	0	888
	KLEIN	1528	19.1	13	[766]	[18]	[134]
64/128	SIMON	1000	16.7	17	282	0	515
	SPECK	1127	13.8	12	186	0	855
	PICCOLO	1334	12.1	9	-	-	-
	PRESENT	1339	12.1	9	[487]	[0]	[96]
96/96	SIMON	984	14.8	15	454	0	454
	SPECK	1134	13.8	12	276	0	866
	EPCBC	1333	12.1	9	[730]	0	[93]
128/128	SIMON	1317	22.9	17	732	0	342
	SPECK	1396	12.1	9	396	0	768
	AES	2400	56.6	24	943	33	445

References

1. Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The simon and speck families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404, 2013. http://eprint.iacr.org/.
2. Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. Simon and speck: Block ciphers for the internet of things. Cryptology ePrint Archive, Report 2015/585, 2015. http://eprint.iacr.org/.
3. Christophe Cannière, Orr Dunkelman, and Miroslav Knežević. Katan and ktantan – a family of small and efficient hardware-oriented block ciphers. In *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems*, pages 272–288. Springer-Verlag, 2009.
4. Frank-Michael Quedenfeld. Modellbildung in der algebraischen kryptoanalyse. 2015.
5. Huaifeng Chen and Xiaoyun Wang. Improved linear hull attack on round-reduced simon with dynamic key-guessing techniques. Cryptology ePrint Archive, Report 2015/666, 2015. http://eprint.iacr.org/.
6. Ning Wang, Xiaoyun Wang, Keting Jia, and Jingyuan Zhao. Differential attacks on reduced simon versions with dynamic key-guessing techniques. Cryptology ePrint Archive, Report 2014/448, 2014. http://eprint.iacr.org/.
7. I. Dinu. Improved differential cryptanalysis of round-reduced speck. In *Selected Areas in Cryptography*, pages 147–164. Springer-Verlag, 2014.
8. S.Lucks F.Abed, E.List and J.Wenzel. Differential cryptanalysis of round-reduced simon and speck. Fast Software Encryption, FSE 2014, LNCS. Springer, 2014.
9. Andrey Bogdanov and Christian Rechbergerv. A 3-subset meet-in-themiddle attack: Cryptanalysis of the lightweight block cipher ktantan. In *Selected Areas in Cryptography*, pages 229–240. Springer-Verlag, 2011.
10. Martin R. Albrecht and Gregor Leander. An all-in-one approach to differential cryptanalysis for small block ciphers. In *Selected Areas in Cryptography*, pages 1–15. Springer-Verlag, 2013.
11. Willi Meier Simon Knellwolf and Mar´a Naya-Plasencia. Conditional differential cryptanalysis of trivium and katan. In *Selected Areas in Cryptography*, pages 200–212. Springer-Verlag, 2012.
12. Thomas Fuhr and Brice Minaud. Match box meet in the middle attack against katan. In *Fast Software Encryption*, pages 61–81. Springer-Verlag, 2015.
13. Yu Sasaki Takanori Isobe and Jiageng Chen. Related-key boomerang attacks on katan32/48/64. In *Information Security and Privacy*, pages 268–285. Springer-Verlag, 2013.
14. Gregory V. Bard Nicolas Courtois Jorge Nakahara Jr. Pouyan Sepehrdad and Bingsheng Zhang. Algebraic, aida/cube and side channel analysis of katan family of block ciphers. In *Progress in Cryptology - INDOCRYPT*, pages 176–196. Springer-Verlag, 2010.