CHAM: A Family of Lightweight Block Ciphers for Resource-Constrained Devices

Bonwook Koo, Dongyoung Roh, Hyeonjin Kim, Younghoon Jung,
Dong-Geon Lee, and Daesung Kwon

National Security Research Institute, Daejeon, Republic of Korea
{bwkoo, dyroh, mikjh, sky1236, guneez, ds_kwon}@nsr.re.kr

Abstract. In this paper, we propose a family of lightweight block ciphers CHAM that has remarkable efficiency on resource-constrained devices. The family consists of three ciphers, CHAM-64/128, CHAM-128/128, and CHAM-128/256 which are of the generalized 4-branch Feistel structure based on ARX(Addition, Rotation, XOR) operations.
In hardware implementations, CHAM requires smaller areas (73% on average) than SIMON [8] through the use of a *stateless-on-the-fly* key schedule which does not require updating a key state. Regarding software performance, it achieves outstanding figures on typical IoT platforms in terms of the balanced performance metrics introduced in earlier works. It shows a level of performance competitive to SPECK [8] mainly due to small memory size required for round keys. According to our cryptanalysis results, CHAM is secure against known attacks.

Keywords: lightweight block cipher, stateless-on-the-fly, ARX

1 Introduction

We are in the pervasive computing era. One can interact with many computing devices, such as laptop computers, tablets, smart phone, and even glasses with computing capabilities. Gartner, Inc. forecasts that in 2017, up 31 percent from 2016, 8.4 billion connected devices will be in use worldwide, with the number reaching 20.4 billion by 2020. The development and spread of these computing devices have made human life more convenient and enriching. However, in terms of adverse effects, threats that existed in cyberspace have spread and expanded to our everyday lives. So, it has become crucial to address security issues in relation to highly constrained devices so as to protect our lives and property. The first step in securing pervasive devices is to ensure that all data from them are transferred confidentially. Cryptographic algorithms, especially block ciphers, have been used extensively to perform this role.

Lightweight cryptography is essential for reducing size and cost of computing and communicating devices. So in cryptography, lightweightness has been moved from implementation issues to design consideration. A lot of block ciphers have been designed for being lightweight in various aspect (for example chip size, code size, power consumption, memory usage, and so on) on various platforms

for the past decade. Some of these algorithms have been shunned for various reasons including security problems, but still many algorithms are being discussed, applied or standardized in the real world such as HIGHT [34], PRESENT [6], CLEFIA [48], KATAN/KTANTAN [22], PRINTCIPHER [38], LED [32], PICCOLO [47], PRINCE [20], SIMON/SPECK [8], LEA [33], PRIDE [1], MIDORI [3], SPARX [29], SKINNY [11] and recently proposed GIFT [4].

Among these algorithms, SIMON and SPECK designed by NSA in 2013, are the most focused and evaluated ciphers. They are families of non-S-box based lightweight block ciphers. SIMON is tuned for optimal performance in hardware, while SPECK is designed for optimal performance in software. SPECK is in ARX structure. They are arguably regarded as the best algorithms among previously proposed lightweight block ciphers on hardware and software platforms.

LEA is another ARX cipher designed by Hong et al. in 2013 and suitable for common platforms (mainly 32-bit microcontrollers), but its performance is not so impressive on 8-bit and 16-bit microcontrollers.

Contributions. We have attempted to improve LEA by increasing the level of suitability for resource-constrained environment such as hardware and 8-bit and 16-bit microcontrollers. As a result, we propose a new family of block ciphers, CHAM.

CHAM has the following features:

- CHAM uses an extremely simple key schedule possibly being implemented without updating key states. This feature helps to reduce the number of flip-flops when implementing our ciphers on hardware.
- Numbers of round keys are far fewer than the numbers of rounds, and round functions reuse them iteratively. This reduces the memory size necessary to store the round keys.
- Encryption uses two types of left rotation, by 1 bit and by 8 bits, to minimize the number of operations on 8-bit AVR microcontroller.
- Round indexes are used as round constants instead of random values to save resources while defending against slide attacks [19] and basic rotational attacks [37].

There are many lightweight block ciphers which have the same feature of key schedule, such as HIGHT, KATAN, PRINTCIPHER, LED, PICCOLO, PRINCE, PICARO [45], PRIDE, and KHUDRA [40]. However, some of these ciphers are prone to or fully broken by related-key attacks [41, 24, 36, 23, 26, 58]. Bearing this in mind, we analyze the security of CHAM against various attacks, including differential cryptanalysis and linear cryptanalysis. In particular, we give more attention to security analyses in the related-key model.

By efficiency evaluation and comparison, we draw the following results. On hardware, we implement SIMON and our ciphers by minimizing the area with the IBM 130nm CMOS generic process library. The results are briefly presented in Table 1. Table 2 shows a brief comparison with SPECK on the two widely adopted target platforms of Atmega128 and MSP430. CHAM shows efficiency competitive with one of the top-ranked algorithms, SPECK on the platforms.

In summary, CHAM can be implemented using smaller area than SIMON in hardware environment and shows a similar level of efficiency to SPECK in 8- and 16-bit software environments.

Table 1. Area-optimized implementation results in gate for CHAM and SIMON.

algorithm	**library**	64/128	128/128	128/256	**ref.**
CHAM	IBM 130nm	665	1,057	1,180	this paper
SIMON	IBM 130nm	958	1,234	1,782	[8]

Table 2. Software efficiency comparison in *rank* metric [9](Larger is better).

scenario	**algorithm**	**Atmega128**	**MSP430**	**ref.**
fixed key [9]	CHAM-64/128	27.9	49.3	this paper
	SPECK-64/128	29.8	50.0	[8]
	CHAM-128/128	17.1	25.0	this paper
	SPECK-128/128	12.7	21.7	[8]
communication [28, 27] (without decryption)	CHAM-64/128	7.2	11.1	this paper
	SPECK-64/128	6.3	9.7	FELICS website

2 Specifications and Design Principles

CHAM is a family of block ciphers with a 4-branch generalized Feistel structure. Each cipher is denoted by CHAM-n/k with a block size of n-bit and a key size of k-bit. Table 3 shows the list of ciphers in the family and their parameters. Here, r and w denote the number of rounds and the bit length of a branch (word), respectively.

Table 3. List of CHAM ciphers and their parameters.

cipher	n	k	r	w	k/w
CHAM-64/128	64	128	80	16	8
CHAM-128/128	128	128	80	32	4
CHAM-128/256	128	256	96	32	8

CHAM consists of three algorithms with different parameters. All three algorithms in the family are suitable for resource-constrained environments, but at the same time they have slightly different applications according to their

parameters. CHAM-64/128 is more suitable for low-end devices, CHAM-128/128 is a better choice for general use on 32-bit microcontrollers, and CHAM-128/256 can serve where a higher security level is required.

2.1 Notations

We use the following notations to describe CHAM-n/k.

- $x \| y$: concatenation of bit strings x and y
- $x \boxplus y$: addition of x and y modulo 2^w
- $x \oplus y$: bit-wise exclusive OR (XOR) of x and y
- $\mathrm{ROL}_i(x)$: rotation of w-bit word x to the left by i bits
- $w_H(x)$: Hamming weight of x, the number of nonzero bits in x.

2.2 Specifications

CHAM-n/k encrypts a plaintext $P \in \{0,1\}^n$ to a ciphertext $C \in \{0,1\}^n$ using a secret key $K \in \{0,1\}^k$ by applying r iterations of the round function as follows.

Divide the plaintext P into four w-bit words $X_0[0]$, $X_0[1]$, $X_0[2]$, and $X_0[3]$, where $P = X_0[0] \| X_0[1] \| X_0[2] \| X_0[3]$. Next, compute the n-bit output $X_{i+1} = X_{i+1}[0] \| X_{i+1}[1] \| X_{i+1}[2] \| X_{i+1}[3]$ of the i-th round for $0 \le i < r$ by

$$\begin{aligned} X_{i+1}[3] &\leftarrow \mathrm{ROL}_8((X_i[0] \oplus i) \boxplus (\mathrm{ROL}_1(X_i[1]) \oplus RK[i \bmod 2k/w])), \\ X_{i+1}[j] &\leftarrow X_i[j+1] \text{ for } 0 \le j \le 2, \end{aligned}$$

if i is even, otherwise,

$$\begin{aligned} X_{i+1}[3] &\leftarrow \mathrm{ROL}_1((X_i[0] \oplus i) \boxplus (\mathrm{ROL}_8(X_i[1]) \oplus RK[i \bmod 2k/w])), \\ X_{i+1}[j] &\leftarrow X_i[j+1] \text{ for } 0 \le j \le 2, \end{aligned}$$

where $RK[i \bmod 2k/w]$ is the round key.

Then, the ciphertext C is defined by $C = X_r[0] \| X_r[1] \| X_r[2] \| X_r[3]$.

The key schedule of CHAM-n/k takes the secret key $K \in \{0,1\}^k$ and generates the $2k/w$ w-bit round keys $RK[0], RK[1], \cdots, RK[2k/w-1]$. Initially, divide the secret key into k/w w-bit words $K[0], K[1], \cdots, K[k/w-1]$, where $K = K[0] \| K[1] \| \cdots \| K[k/w-1]$. Then the round keys are generated as follows.

$$\begin{aligned} RK[i] &\leftarrow K[i] \oplus \mathrm{ROL}_1(K[i]) \oplus \mathrm{ROL}_8(K[i]), \\ RK[(i+k/w) \oplus 1] &\leftarrow K[i] \oplus \mathrm{ROL}_1(K[i]) \oplus \mathrm{ROL}_{11}(K[i]), \end{aligned}$$

where $0 \le i < k/w$.

The structures of the key schedule and round function of CHAM are depicted in Fig. 1, and the test vectors are provided in appendix A.

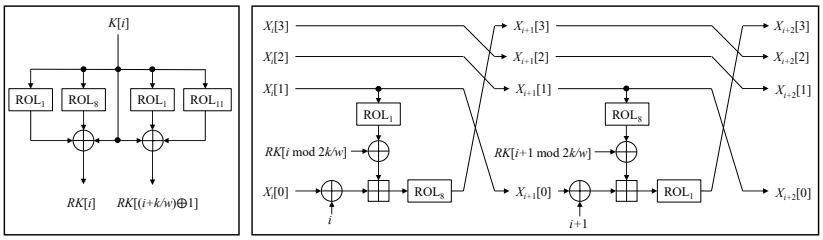

Fig. 1. Key schedule (left) and two consecutive round functions beginning with the even i-th round (right).

2.3 Design Principles

Design paradigm and Design approach Our goal is to design a cipher that shows better efficiency than SIMON and SPECK on both (resource-constrained) hardware and software. So we design in the ARX design paradigm so that our cipher could have advantages in various resource-constrained environments. However, it is still not easy to prove nor to theoretically bound the minimum number of rounds with respect to the security of an ARX block cipher (although there is an exception [29]). Hence, we decided to design our algorithms (round function and key schedule) with a focus on efficiency and lightweight aspect and to perform a comprehensive and detailed cryptanalysis relying on experimental methods.

Round Functions In the instruction sets of AVR and MSP introduced in Section 4, there is no single instruction for a rotation of arbitrary amounts. Only byte-swap and 1-bit rotation can be used for implementing $\mathrm{ROL}_i()$. So, if we want to implement 3-bit rotation, we have to implement 1-bit rotation for 3 times. Hence, we choose every rotation amount as close as possible to 0 or 8 for the round functions (as well as the key schedule) to achieve a better performance. At the same time, in order to reduce the number of rounds while keeping security, we select several combinations of rotation amounts for a round function and consider combinations of the round functions for the whole encryption.

At first, we compare several cases of using a single round function and those of using two slightly different round functions. Let's denote rotation amounts of two consecutive rounds as a 4-tuple (a, b, c, d) such that a and c are rotation amounts for the values before the round key XOR and b and d are rotation amounts for outputs of the additions. For example, CHAM is of type $(1, 8, 8, 1)$. We estimate the maximum numbers of rounds of differential characteristics with probabilities greater than 2^{-n} when $w = 16$. Table 4 shows the results.

In Table 4, the type $(1, 8, 9, 1)$ looks like the best choice, however the type $(1, 8, 9, 1)$ requires more operations than the type $(1, 8, 8, 1)$ for every two rounds, so the overall efficiency of the type $(1, 8, 8, 1)$ is better. Also, we test the case of using 4-round alternating structures but could not find any better case than our selection in the overall efficiency point of view.

Table 4. Security evaluation results of several types of round functions.

type	$(0,9,0,9)$	$(1,8,1,8)$	$(1,8,1,9)$	$(1,8,8,1)$	$(1,8,9,1)$
diff. ch. round	58	47	40	36	33

Additionally, we use round indexes as round constants not only to defend the slide attack and the rotational attack and but also to reduce the extra registers for storing constants on both hardware and software implementations.

Key Schedule Key schedule is a procedure to calculate round keys from a given secret key. Many block ciphers (AES, SIMON/SPECK [8], and so on) have their own key schedules that consist of functions calculating each round keys from the current key state and update the key state. And key schedules of some other ciphers (PICCOLO [47], LED [32], and so on) consist of functions which calculate every round key directly from the secret key. Although the former key schedules could be also implemented to calculate every round key directly from the secret key, the latter key schedules are more suitable for such implementation. We call this implementation method as *stateless-on-the-fly* implementation and a key schedule like latter ones as *stateless-on-the-fly* key schedule.

It is clear that a stateless-on-the-fly implementation of a key schedule does not require areas (flip-flops) for storing key states on hardware, so it can be replaced by some selecting logics. For a k-bit secret key, at least $k \times s_f$ gates are required for storing key state while roughly at most $(k-1) \times s_m$ gates are needed for some selecting logic, where s_f and s_m are gate counts of D flip-flop and 2-to-1 MUX, respectively. Therefore, the stateless-on-the-fly implementation helps to save the hardware areas when $s_m < s_f$. For an example of CHAM-64/128 in IBM130 [8], we can theoretically save 259 gates by adopting stateless-on-the-fly implementation for serial implementation, since $s_m = 2.25$ and $s_f = 4.25$. In practice, more gates could be saved for stateless-on-the-fly implementation because compiler could employ 3-to-1 or 4-to-1 MUXes instead of 2-to-1 MUXes to optimize the area while D flip-flop cannot be replaced by the other cell.

The key schedules of CHAM consist of a linear function Φ from $\{0,1\}^w$ to $\{0,1\}^{2w}$. Let I_w be the $w \times w$ identity matrix and I_w^l be a $w \times w$ matrix whose i-th row is defined by l times left rotation of i-th row of I_w for $i = 1, 2, ..., w$. Then, the function Φ can be expressed by multiplication with the following binary matrix A defined by

$$A = (A_1|A_2)^T = \left(I_w \oplus I_w^1 \oplus I_w^8 \mid I_w \oplus I_w^1 \oplus I_w^{11}\right)^T.$$

Let $\mathcal{A}$ be a set of all matrices $\left(I_w^{l_0} \oplus I_w^{l_1} \oplus I_w^{l_2} \mid I_w^{l_3} \oplus I_w^{l_4} \oplus I_w^{l_5}\right)^T$, where $l_0 = l_3 = 0, l_0 \neq l_1 \neq l_2, l_3 \neq l_4 \neq l_5$ and $0 < l_1, l_2, l_4, l_5 < w$. Let

$$\mathcal{A}_j = \{M \in \mathcal{A} | \min_{x \in \{0,1\}^w \setminus \{\mathbf{0}\}} (w_H(M \cdot x)) = j\}.$$

We make certain that in the case of $w = 16$, $\mathcal{A} = \mathcal{A}_2 \cup \mathcal{A}_4 \cup \mathcal{A}_6$.

The matrix A is the case of $l_0 = 0$, $l_1 = 1$, $l_2 = 8$, $l_3 = 0$, $l_4 = 1$, and $l_5 = 11$. It is chosen from $\mathcal{A}_6$ considering the number of instructions for implementations on AVR and MSP and cryptographic properties in the related-key model. Note that the matrix with $l_0 = 0$, $l_1 = 1$, $l_2 = 8$, $l_3 = 0$, $l_4 = 1$, and $l_5 = 9$ is in $\mathcal{A}_4$ and shows worse property for the related-key differential cryptanalysis. And the matrix with $l_0 = 0$, $l_1 = 1$, $l_2 = 8$, $l_3 = 0$, $l_4 = 1$, and $l_5 = 10$ also shows worse property than A in the viewpoint of the related-key differential cryptanalysis, even though it belongs $\mathcal{A}_6$.

Additionally, we let the round keys be iteratively re-applied to reduce the size of memory for storing the round keys in software. And we give the round key index a tweak so that a w-bit word of a secret key cannot regularly act on every k/w rounds.

3 Hardware Implementation

We implemented CHAM in Verilog and synthesized using Synopsys Design Compiler 2008.09-SP5 without the *compile_ultra* feature. For the synthesis, the 100-kHz frequency constraint is imposed. Clocks for plaintext flip-flops are gated and scan flip-flops are not used. To evaluate and compare the hardware cost and performance, we use the UMC 90nm, UMC 180nm CMOS generic process library (UMC90, UMC180) provided by the Faraday Technology Corporation and IBM's 8RF 130nm process (IBM130).

Figs. 2 and 3 present the round-based and the bit-serial hardware architecture of CHAM, respectively. In both architectures, there are no flip-flops for storing the secret key because the secret key is injected for each clock count. Instead of the flip-flops, a k/w-to-1 w-bit MUX is required for the round-based architecture, and a k-to-1 1-bit MUX is required for the bit-serial architecture. Since the size of the flip-flops is larger than that of the MUXes, we can save as much as the difference between the two sizes.

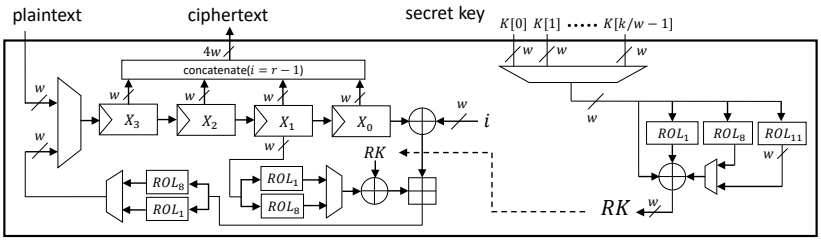

Fig. 2. Round-based hardware architecture of CHAM.

Table 5 shows the results of comparison to other lightweight block ciphers for each parameter and architecture in area (GE, gate equivalent) and throughput

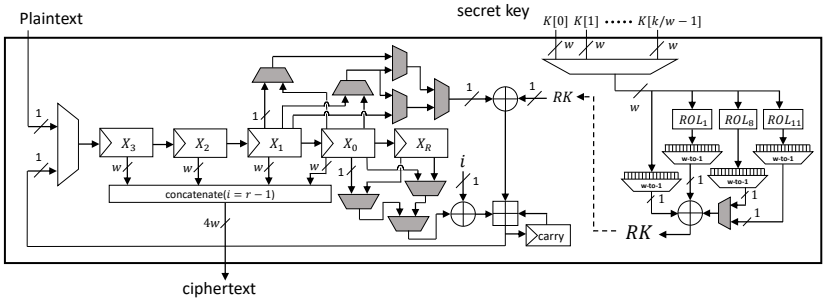

Fig. 3. Serialized hardware architecture of CHAM.

(Kbps@100KHz). As the table shows, every instance in our family can be implemented in much smaller area than SIMON with the corresponding parameter. Precisely, the area of the sequential logic for CHAM is much smaller than that of SIMON because the secret key is not stored in flip-flops. Moreover, in most cases (except for the round-based CHAM-64/128), the throughputs per area's are also better than those of SIMON.

4 Software Implementation

We compare software performances of our cipher and existing algorithms on three typical microcontrollers suitable for lightweight devices. In [10, 27, 57], Atmega128(AVR), MSP430F1611(MSP), and SAM3X8E(ARM) are widely used as target 8, 16, and 32-bit microcontrollers, respectively. We also choose them as our target platforms. Their features are briefly presented in Appendix B.1.

Software performances are measured based on two metrics, *rank* and FOM, under two usage scenarios, *Fixed-key* and *Communication*. The metrics and scenarios are introduced in [9, 27, 28], and described in Appendices B.3, B.4.

All our software implementations of CHAM are done in assembly language to draw better performance as much as possible. Some of the ideas are presented in Appendix B.2. Note also that, in our work, all the best results of CHAM are obtained from 4-round unrolled implementations except for CHAM-128/128 on the ARM. In that case, 8-round unroll results the best FOM value.

Table 6 presents the results of a performance comparison on the AVR and MSP platforms using the rank metric for the fixed-key scenario. Either SPECK or CHAM is always ranked at the top on both AVR and MSP, for all of the parameters of the block size and key size. They outperform the remaining ciphers. The results for SPARX [29] and LEA are drawn from performance reports posted on the FELICS [27] project's website.

Table 7 shows the performances based on the rank metric under the one-way communication scenario, emphasizing variable key encryption without decryption.

Table 5. Comparison of the hardware implementation of CHAM and other ciphers.

n	k	cipher	bit-serial		round-based		tech.	ref.
			area (GE)	through-ghput	area (GE)	through-ghput		
64	128	Twine			1,866	178.0	90n	[51]
		Hight			3,048	188.3	250n	[34]
		Gift	930		1,345		STM90	[4]
		Midori			1,542		STM90	[3]
		Midori			1,638		STM65	[3]
		Skinny	1,399	8.1	1,696	177.8	UMC180	[11]
		LED	1,265	3.4				[32]
		†LED	700	3.4				[32]
		Present	1,391	11.4	1,884	200.0	UMC180	[46]
		Piccolo	758	12.1	1,197	193.9	130n	[47]
		Simon	944	4.2	1,403	133.3	‡IBM130	[59]
		Simeck	924	4.2	1,365	133.3	‡IBM130	[59]
		Simon	958	4.2	1,417	133.3	IBM130	[8]
		Speck	996	3.4	1,658	206.5	IBM130	[8]
		CHAM	665	5.0	826	80.0	IBM130	ours
		CHAM	859	5.0	1,110	80.0	UMC180	ours
		CHAM	727	5.0	985	80.0	UMC90	ours
128	128	Gift	1,213		1,997		STM90	[4]
		Midori			2,522		STM90	[3]
		Midori			2,714		STM65	[3]
		Skinny	1,840	14.7	2,391	320.0	UMC180	[11]
		LEA	2,302	4.2	3,826	76.2	UMC130	[33]
		AES			2,400	57.0	UMC180	[44]
		Simon	1,234	2.9	2,090	182.9	IBM130	[8]
		Speck	1,280	3.0	2,727	376.5	IBM130	[8]
		CHAM	1,057	5.0	1,499	160.0	IBM130	ours
		CHAM	1,296	5.0	1,899	160.0	UMC180	ours
		CHAM	1,084	5.0	1,691	160.0	UMC90	ours
128	256	Skinny	2,655	12.3	3,312	266.7	UMC180	[11]
		Simon	1,782	2.6	2,776	168.4	IBM130	[8]
		Speck	1,840	2.8	3,284	336.8	IBM130	[8]
		CHAM	1,180	4.2	1,622	133.3	IBM130	ours
		CHAM	1,481	4.2	2,087	133.3	UMC180	ours
		CHAM	1,256	4.2	1,864	133.3	UMC90	ours

†: Hard-wired key implementation

‡: Not exactly same to the non-daggered IBM130

Table 6. Performance comparison using the rank metric on AVR and MSP, under the fixed-key scenario.

size(n/k)	algorithm	AVR				MSP			
		ROM	**RAM**	**cpb**	**rank**	**ROM**	**RAM**	**cpb**	**rank**
64/128	SPECK[10]	218	0	154	29.8	204	0	98	50.0
	CHAM	202	3	172	27.9	156	8	118	49.3
	SIMON[10]	290	0	253	13.6	280	0	177	20.2
	SPARX	448	2	224	9.9	366	14	136	18.7
	HIGHT[9]	336	0	311	9.6	-	-	-	
128/128	CHAM	362	16	148	17.1	280	20	125	25.0
	SPECK[10]	460	0	171	12.7	438	0	105	21.7
	AES[10]	970	18	146	6.8	-	-	-	-
	LEA	754	17	203	6.3	646	24	147	9.8
128/256	CHAM	396	16	177	13.2	312	20	148	19.2
	SPECK[9]	476	0	181	11.6	-	-	-	-

The results of CHAM on the AVR and MSP, obtained by optimization[1] level two, show slightly better performance than that of SPECK. The ranks of SPECK are derived from the data reported in the FELICS website, and the best ones are shown in the table.

On the ARM platform, CHAM-64/128 shows relatively poor result due to its 16-bit word size.[2] On the other hand, CHAM-128/128, based on a 32-bit word size, presents relatively decent result compared to SPECK-64/128.[3]

Table 8 shows performance comparison based on the FOM metric under the communication scenario. SPECK-64/128 reveals the best value, followed by CHAM-128/128.[4] Considering its block size, CHAM-128/128 can be regarded as a very good performer. The FOM of CHAM-64/128 is degraded by relatively poor performance on ARM. However, due to the excellence on AVR and MSP, its FOM is still ahead of all other competitors, except for SPECK. Note that all the values other than CHAM are presented on the FELICS website.

[1] The FELICS platform provides a unified implementation environment which generates performance figures automatically. The FELICS software framework, written in C language, permits users to implement only the core parts of encryption, decryption and their key schedules. Due to the common operational C source codes, the performance results are affected by the compiler's optimization option.

[2] A SIMD implementation might enhance performance. Since the ARMv7-M architecture provides very limited instructions for SIMD arithmetics, it seems to be very difficult to get non-trivial performance gain from SIMD approach.

[3] The performance of SPECK-128/128 is not yet reported in the FELICS website.

[4] In the comparison, we exclude Chaskey algorithm because it is not considered as a block cipher.

Table 7. Performance comparison using the rank metric in the one-way communication scenario. EKS and Enc represent the key schedule and encryption.

platform	algorithm	ROM		RAM		cycles		cpb	rank
		EKS	Enc	stack	data	EKS	Enc		
AVR	CHAM-64/128	72	280	11	184	309	23,664	187	7.2
	SPECK-64/128	178	240	12	260	1,401	19,888	166	6.3
MSP	CHAM-64/128	68	210	16	184	275	16,715	133	11.1
	SPECK-64/128	126	180	16	260	1,242	14,155	120	9.7
ARM	SPECK-64/128	52	164	36	260	516	6,323	53	23.2
	CHAM-128/128	44	210	48	192	95	8,846	70	19.5
	CHAM-64/128	124	272	48	184	170	16,944	134	8.7

Table 8. Performance comparison using the FOM metric in the communication scenario. The key schedule, encryption and decryption steps are all included.

algorithm	AVR			MSP			ARM			FOM
	ROM	RAM	cpb	ROM	RAM	cpb	ROM	RAM	cpb	
SPECK-64/128	874	302	351	572	296	253	444	308	129	5.00
CHAM-128/128	1,230	262	337	926	258	298	528	272	144	5.47
CHAM-64/128	844	225	394	578	220	290	606	244	319	6.52
RECTANGLE-64/128	1,118	353	506	844	402	361	654	432	289	7.80
SPARX-64/128	1,198	392	512	966	392	287	1,200	424	319	8.57

5 Security Analysis

In this section, we summarize the security analysis results of CHAM and describe how we determine the number of rounds for each algorithm. We then present detailed cryptanalysis results of a number of well-known attacks applicable to our ciphers.

5.1 Summary of the Security Analysis

Table 9 shows the maximum numbers of rounds of characteristics that can be used for each attack. Based on these results, we determine the total number of rounds of each cipher with the following considerations

- Round extension of characteristics
- Round extension during key-recovery phase
- Security margin

By applying the probability gathering technique used in [49] to our differential cryptanalysis, an attacker may find differentials longer than the differential characteristics in Table 10 in appendix C.1. But we check experimentally and confirm that the attacker can have at most four rounds longer differentials. Also, we verified that a boomerang and a rectangle characteristics can also be extended at most four rounds despite their use of two differential characteristics. Although

Table 9. The numbers of rounds of the best discovered characteristics for each cipher and cryptanalysis.

CHAM-	DC	RDC	LC	BC	RBC	IDC	ZCLC	DLC	RDLC	IC	RXDC
64/128	36	34	34	35	**41**	18	21	34	36	16	16
128/128	45	33	40	**47**	36	15	18	45	39	16	23
128/256	45	40	40	**47**	**47**	15	18	45	45	16	23

(R)DC: (Related-key)Differential Cryptanalysis, LC: Linear Cryptanalysis, (R)BC: (Related-key)Boomerang Cryptanalysis, IDC: Impossible Differential Cryptanalysis ZCLC: Zero-Correlation Linear Cryptanalysis, (R)DLC: (Related-key)Differential-Linear Cryptanalysis, IC: Integral Cryptanalysis, RXDC: Rotational-XOR-Differential Cryptanalysis

such extensions can be applied only to the cryptanalyses based on difference or linear approximation, we assume every type of characteristic can be extended by four rounds regarding this technique.

In the key-recovery phase of an attack, if $k = 2n$, then one can attack k/w more rounds by guessing $k/2$ bits of the secret key before filtering. Otherwise, one cannot utilize this type of round extension. After filtering, one can add at most three rounds during key guessing and determining steps. Hence, we claim that the maximal numbers of rounds for which one can mount a key-recovery attack are at most $4 + 2(k/w - 4) + 3$ more than the numbers of rounds in Table 9, regardless of actual characteristics.

To estimate the numbers of rounds conservatively, we assume that an attacker can mount successful attacks using the characteristics we have found, even if the actual attacks are infeasible. Finally, we determine the numbers of rounds (instances of r), as shown in Table 3 with these considerations together with a security margin greater than 30% for full rounds[5].

5.2 Cryptanalysis Results

We cryptanalyze our ciphers with the following well-studied techniques, and the results are summarized in Table 9. We use state-of-the-art techniques, modified for ARX structure ciphers [49, 17, 7, 53]. Using these results, we estimate the necessary numbers of rounds to ensure proper security. This subsection includes the results of (RK[6]) Differential, Linear, and (RK) Boomerang cryptanalysis and other detailed cryptanalysis results are provided in Appendix C due to the page limit.

(RK) Differential and Linear Cryptanalysis. We searched for (RK) differential characteristics [16] and linear approximations of each cipher using an automated algorithm known as the *threshold search* suggested in [17]. It is considered as an appropriate variant of Matsui's branch-and-bound algorithm [43] for ARX ciphers. Using this threshold search approach, we found differential

[5] 30% is a relatively high ratio for a security margin compared to those associated with other ciphers.

[6] RK stands for "related-key".

characteristics with a probability of $p > 2^{-n}$ for each cipher. These results are given in Table 10. We denote a (RK) differential characteristic by (Δ_{key}), $\Delta_{\text{in}} \rightarrow \Delta_{\text{out}}$, where Δ_{key} is the difference of the secret key, and Δ_{in} and Δ_{out} are the input and output differences, respectively.

For a linear cryptanalysis [42], we found linear approximations with bias $\epsilon > 2^{-n/2}$ for each cipher. These results are given in Table 11. Γ_{in} and Γ_{out} denote the input and output masks, respectively. We calculate the correlation of the linear approximation by determining the bias of every addition using Wallén's formula [56] and applying Piling-Up Lemma [42].

(RK) Boomerang Cryptanalysis. The (RK) boomerang characteristics [54, 15] are constructed with two short (RK) differential characteristics. These characteristics are the most threatening ones for our ciphers. Examples of short (RK) differential characteristics for constructing (RK) boomerang characteristics are shown in Table 12. A (RK) boomerang characteristic is valid when $pq > 2^{-n/2}$, where p and q are the probabilities of the two differential characteristics ϕ_1 and ϕ_2, respectively, used for constructing the (RK) boomerang characteristic. Note that (RK) rectangle cryptanalysis [13, 15] works using the same characteristics.

6 Conclusion

In this paper, we have presented CHAM, a family of lightweight block ciphers that supports widely used block/key lengths. It is suitable for highly resource-constrained devices, especially area-constrained hardware with the help of the *stateless-on-the-fly* key schedule. Efficiency evaluations and comparisons with other lightweight block ciphers on various platforms are provided to demonstrate the merit of our algorithms. A variety of security analyses with extensive searching have convinced us that it is secure against known attacks. We also believe that it will be resistant to future attacks due to its high security margins. Moreover, the simplicity and flexibility of CHAM allow one to design ciphers with other block sizes and key lengths. Of course, a rigorous security analysis should be followed.

References

1. Albrecht, M.R., Driessen, B., Kavun, E.B., Leander, G., Paar, C., Yalçın, T.: Block ciphers focus on the linear layer (feat. PRIDE). In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 57-76. Springer, Heidelberg (2014)
2. Ashur, T., Liu, Y.: Rotational cryptanalysis in the presence of constants. IACR Trans. Symmetric Cryptol., 2016(1):57-70, 2016.
3. Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: A block cipher for low energy. In Iwata, T., Cheon, J.H., (eds.) ASIACRYPT 2015, Part II. LNCS, vol. 9453, pp. 411-436. Springer, Heidelberg (2015)
4. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: A Small PRESENT. to appear in Cryptographic Hardware and Embedded Systems - CHES 2017 - Taipei, Taiwan, September 25-28, 2017

5. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique Cryptanalysis of the Full AES. In: Lee, D.H. (ed.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 344-371. Springer, Heidelberg (2011)
6. 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)
7. Bogdanov, A., Rijmen, V. (2014) Linear hulls with correlation zero and linear cryptanalysis of block ciphers. Des. Codes Cryptography 70, 3 (March 2014), 369-383. DOI=http://dx.doi.org/10.1007/s10623-012-9697-z
8. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L. (2013) The Simon and Speck Families of Lightweight Block Ciphers. IACR Cryptology ePrint Archive (2013):404
9. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The Simon and Speck block ciphers on AVR 8-bit microcontrollers. In: Eisenbarth, T., Öztürk, E. (eds.) LightSec 2014. LNCS, vol. 8898, pp. 3-20. Springer, Heidelberg (2015)
10. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L. (2015) Simon and Speck: Block ciphers for the Internet of things. IACR Cryptology ePrint Archive (2015): 585
11. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 123-153, Springer, Heidelberg (2016)
12. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 12-23. Springer, Heidelberg (1999)
13. Biham, E., Dunkelman, O., Keller, N.: The rectangle attack - rectangling the Serpent. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 340-357. Springer, Heidelberg (2001)
14. Biham, E., Dunkelman, O., Keller, N.: Enhancing differential-linear cryptanalysis. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 587-592. Springer, Heidelberg (2002)
15. Biham, E., Dunkelman, O., Keller, N.: Related-key boomerang and rectangle attacks. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 507-525. Springer, Heidelberg (2005)
16. Biham, E., Shamir, A. (1993) Differential Cryptanalysis of the Data Encryption Standard, Springer, DOI: 10.1007/978-1-4613-9314-6 ISBN: 978-1-4613-9316-0, 978-1-4613-9314-6 Springer New York
17. Biryukov, A., Velichkov, V.: Automatic Search for Differential Trails in ARX Ciphers. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 227-250. Springer, Heidelberg (2014)
18. Biryukov, A., Cannière, C.D., Quisquater, M.: On Multiple Linear Approximations. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 1-22. Springer, Heidelberg (2004)
19. Biryukov, A., Wagner, D.: Slide attacks. In: Knudsen, L. (ed.) FSE 1999. LNCS, vol. 1636, pp. 245-259. Springer, Heidelberg (1999)
20. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçın, T.: PRINCE A Low-Latency Block Cipher for Pervasive Computing

Applications. In: Wang X., Sako K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208-225. Springer, Heidelberg (2012)
21. Buhrow, B., Riemer, P., Shea, M., Gilbert, B., Daniel, E.: Block Cipher Speed and Energy Efficiency Records on the MSP430: System Design Trade-Offs for 16-Bit Embedded Applications. In: Aranha D., Menezes A. (eds.) LATINCRYPT 2014. LNCS, vol. 8895, pp. 104-123. Springer, Heidelberg (2015)
22. 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)
23. Canteaut, A., Lallemand, V., Naya-Plasencia, M.: Related-Key Attack on Full-Round PICARO. In: Dunkelman O., Keliher L. (eds.) SAC 2015. LNCS, vol. 9566, pp. 86-101. Springer, Heidelberg (2016)
24. Chen, J., Teh, J.S., Su, C., Samsudin, A., Fang, J.: Improved (related-key) Attacks on Round-Reduced KATAN-32/48/64 Based on the Extended Boomerang Framework. In Liu J., Steinfeld R. (eds.) ACISP 2016. LNCS, vol. 9723, pp. 333-346. Springer, Heidelberg (2016)
25. Courtois, N.T., Pieprzyk, J.: Cryptanalysis of block ciphers with overdefined systems of equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267-287. Springer, Heidelberg (2002)
26. Dai, Y., Chen, S.: Cryptanalysis of full PRIDE block cipher. Sci. China Inf. Sci. 60: 052108. doi:10.1007/s11432-015-5487-3 (2017)
27. Dinu, D., Biryukov, A., Großschädl, J., Khovratovich, D., Le Corre, Y., Perrin, L. FELICS – Fair Evaluation of Lightweight Cryptographic Systems In: NIST Workshop on Lightweight Cryptography 2015 National Institute of Standards and Technology (2015)
28. Dinu, D., Le Corre, Y., Khovratovich, D., Perrin, L., Großschädl, J., Biryukov, A. Triathlon of lightweight block ciphers for the Internet of things IACR Cryptology ePrint Archive (2015): 209
29. Dinu, D., Perrin, L., Udovenko, A., Velichkov, V., Großschädl, J., Biryukov, A.: Design strategies for ARX with provable bounds: Sparx and LAX. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 484-513. Springer, Heidelberg (2016)
30. ECRYPT II, Implementations of low cost block ciphers in Atmel AVR devices, `http://perso.uclouvain.be/fstandae/lightweight_ciphers/`.
31. Eisenbarth, T., Gong, Z., Güneysu, T., Heyse, S., Indesteege, S., Kerckhof, S., Koeune, F., Nad, T., Plos, T., Regazzoni, F., Standaert, F.-X., van Oldeneel tot Oldenzeel, L.: Compact Implementation and Performance Evaluation of Block Ciphers in ATtiny Devices. In: Mitrokotsa, A., Vaudenay, S. (eds.) AFRICACRYPT 2012. LNCS, vol. 7374, pp. 172-187. Springer, Heidelberg (2012)
32. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.J.B.: The LED Block Cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326-341. Springer, Heidelberg (2011)
33. Hong, D., Lee, J.-K., Kim, D.-C., Kwon, D., Ryu, K.H., Lee, D.-G.: LEA: a 128-bit block cipher for fast encryption on common processors. In: Kim, Y., Lee, H., Perrig, A. (eds.) WISA 2013. LNCS, vol. 8267, pp. 3-27. Springer, Heidelberg (2014).
34. 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)

35. Huang, J., Vaudenay, S., Lai, X., Nyberg, K.: Capacity and data complexity in multidimensional linear attack. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 141-160. Springer, Heidelberg (2015)
36. Jean, J., Nikolić, I., Peyrin, T., Wang, L., Wu, S.: Security analysis of PRINCE. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 92-111. Springer, Heidelberg (2014)
37. Khovratovich, D., Nikolić, I.: Rotational cryptanalysis of ARX. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 333-346. Springer, Heidelberg (2010)
38. Knudsen, L.R., Leander, G., Poschmann, A., Robshaw, M.J.B.: PRINTcipher: A block cipher for IC-printing. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 16-32, Springer, Heidelberg (2010)
39. Knudsen, L.R., Wagner, D.: Integral Cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112-127. Springer, Heidelberg (2002)
40. Kolay, S., Mukhopadhyay, D.: Khudra: A new lightweight block cipher for FPGAs. In: Chakraborty, R.S., Matyas, V., Schaumont, P. (eds.) SPACE 2014. LNCS, vol. 8804, pp. 126-145. Springer, Heidelberg (2014)
41. Koo, B., Hong, D., Kwon, D.: Related-key attack on the full HIGHT. In: Rhee, K.-H., Nyang, D.H. (eds.) ICISC 2010. LNCS, vol. 6829, pp. 49-67. Springer, Heidelberg (2011)
42. Matsui, M.: Linear cryptanalysis of Data Encryption Standard. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386-397. Springer, Heidelberg (1994)
43. Matsui, M.: On correlation between the order of s-boxes and the strength of DES. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 366-375. Springer, Heidelberg (1995)
44. 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)
45. Piret, G., Roche, T., Carlet, C.: PICARO - a block cipher allowing efficient higher-order side-channel resistance. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 311-328. Springer, Heidelberg (2012)
46. Poschmann, A.: Lightweight Cryptography - Cryptographic Engineering for a Pervasive World. Number 8 in IT Security. Europischer Universittsverlag, Published: Ph.D. Thesis, Ruhr University Bochum (2009)
47. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: An Ultra-Lightweight Blockcipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 342-357. Springer, Heidelberg (2011)
48. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-Bit Blockcipher CLEFIA (Extended Abstract). In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 181-195. Springer, Heidelberg (2007)
49. Song, L., Huang, Z., Yang, Q.: Automatic Differential Analysis of ARX Block Ciphers with Application to SPECK and LEA. In: Liu, J.K., Steinfeld, R. (eds.) ACISP 2016, Part II. LNCS, vol. 9723, pp. 379–394. Springer, Heidelberg (2016)
50. Sun, L., Wang, W., Liu, R., Wang, M. (2016) MILP-Aided Bit-Based Division Property for ARX-Based Block Cipher, Cryptology ePrint Archive, Report 2016:1101
51. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: TWINE: A Lightweight Block Cipher for Multiple Platforms. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 339-354. Springer, Heidelberg (2013)
52. Todo, Y.: Integral Cryptanalysis on Full MISTY1. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 413-432. Springer, Heidelberg (2015)

53. Todo, Y., Morii, M.: Bit-Based Division Property and Application to Simon Family. In: Peyrin T. (ed) FSE 2016. LNCS, vol 9783, pp. 357-377. Springer, Heidelberg (2016)
54. Wagner, D.: The boomerang attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156-170. Springer, Heidelberg (1999)
55. Wang Y., Wu W., Yu X., Zhang L.: Security on LBlock against biclique cryptanalysis. In: Lopez J., Tsudik G. (eds.) WISA 2012. LNCS, vol. 7690, pp. 1-14. Springer, Heidelberg (2012)
56. Wallén J. (2003) On the Differential and Linear Properties of Addition Master's thesis, Helsinki University of Technology, Laboratory for Theoretical Computer Science
57. Wenzel-Benner, C., Gräf, J.: XBX: eXternal Benchmarking eXtension for the SUPERCOP Crypto Benchmarking Framework. In: Mangard, Standaert (eds.) CHES 2010. LNCS, vol. 6225, pp. 294-305. Springer, Heidelberg (2010)
58. Yang, Q., Hu, L., Sun, S., Song, L.: Related-Key Impossible Differential Analysis of Full Khudra. In: Ogawa K., Yoshioka K. (eds.) IWSEC 2016. LNCS, vol. 9836. pp. 135-146. Springer, Heidelberg (2016)
59. Yang, G., Zhu, B., Suder, V., Aagaard, M.D., Gong, G.: The simeck family of lightweight block ciphers. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 307-329. Springer, Heidelberg (2015)

A Test Vectors

Test vectors are represented in hexadecimal with the prefix '0x'.

CHAM-64/128

```
secret Key : 0x0100 0x0302 0x0504 0x0706 0x0908 0x0b0a 0x0d0c 0x0f0e
plaintext  : 0x1100 0x3322 0x5544 0x7766
ciphertext : 0x453c 0x63bc 0xdcfa 0xbf4e
```

CHAM-128/128

```
secret Key : 0x03020100 0x07060504 0x0b0a0908 0x0f0e0d0c
plaintext  : 0x33221100 0x77665544 0xbbaa9988 0xffeeddcc
ciphertext : 0xc3746034 0xb55700c5 0x8d64ec32 0x489332f7
```

CHAM-128/256

```
secret Key : 0x03020100 0x07060504 0x0b0a0908 0x0f0e0d0c
             0xf3f2f1f0 0xf7f6f5f4 0xfbfaf9f8 0xfffefdfc
plaintext  : 0x33221100 0x77665544 0xbbaa9988 0xffeeddcc
ciphertext : 0xa899c8a0 0xc929d55c 0xab670d38 0x0c4f7ac8
```

B Some Details about Software Implementation

B.1 Target Platforms

Atmega128 belongs to Atmel's AVR microcontroller family with an 8-bit RISC architecture. It has 32 general-purpose registers and 133 instructions. It is equipped

with 128 KBytes of flash and 4 KBytes of RAM. MSP430F1611, a microcontroller from Texas Instruments, adopts a 16-bit RISC architecture with 16 registers (12 of them are general-purpose registers) and 51 instructions, including the emulated ones. It has 48 KBytes of flash and 10 KBytes of RAM. The ARM Cortex-M3 is a 32-bit processor core based on the ARMv7-M architecture, with 12 general-purpose registers. The core is adopted in the Atmel SAM3X8E microcontroller installed on the well-known Arduino Due development board. SAM3X8E is equipped with 512 KBytes of flash and 96 KBytes of SRAM.

B.2 Implementating Bit-wise Rotation

In ARX design like CHAM, rotations by certain bit sizes might be costly to implement on our 8 and 16-bit platforms. Efficient implementation of rotation is crucial to both high throughput and smaller memory. With this consideration,CHAM adopts ROL_8, which can be performed for free on AVR platform.

The MSP430 provides a byte-swapping instruction, which is equivalent to ROL_8 for a 16-bit word. It is slightly tricky for a 32-bit word on the MSP430. Similarly to [21], ROL_8 can be carried out in seven instructions, as in Code 1 below.[7] The code require an additional temporary register to hold a 16-bit data.

ARMv7-M provides a powerful instruction, barrel shifter, which can rotate a 32-bit word by any bit-size. Moreover, the instruction can perform a certain kind of operation additionally after the rotation. This fact gives rise to a good performance of CHAM-128/128. However, it appears that no single instruction of ARMv7-M can perform bit-wise rotation for a *16-bit* word. This explains the relatively low performance of CHAM-64/128 on ARMv7-M, as can be seen in Table 7 and 8.

Code 1. MSP430 code for 8-bit left rotation; register pairs `Rh` and `Rl` store a 32-bit integer, `ABCD`, where each letter represents an 8-bit integer part; register `Rt` is temporary.

```
mov     Rl, Rt      :   Rh=(B,A),  Rl=(D,C),  Rt=(D,C)          ABCD
xor.b   Rh, Rl      :              Rl=(B^D,O)
xor     Rh, Rl      :              Rl=(D,A)
xor.b   Rt, Rh      :   Rh=(B^D,O)
xor     Rt, Rh      :   Rh=(B,C)
swpb    Rl          :              Rl=(A,D)                     OODA
swpb    Rh          :   Rh=(C,B)                                BCOO
```

B.3 Performance Metrics

Lightweight IoT devices are usually considered to have constrained resources. This is why throughput alone does not fully describe the performance of an algorithm. A smaller code size and less RAM usage are also important factors to

[7] We implement ROR_8, a right rotation for decryption, in eight instructions.

consider. In [9], the authors argue the same context and introduce the metric of *rank* as an overall performance indicator. It is defined as

$$\text{rank} = (10^6/\text{cpb})/(\text{ROM} + 2 \times \text{RAM}),$$

where *cpb* refers to the cycles per byte consumed for a task, and *ROM* and *RAM* are the byte sizes of the memory of each type. By definition, the larger rank is better. Note also that RAM is considered to be twice as costly as ROM.

The FOM(figure of merit) metric, defined recently in the FELICS [28], averages performances on AVR, MSP and ARM. For each implementation i on a device d, we measure memory usages $v^{i,d}_{\text{ROM}}$, $v^{i,d}_{\text{RAM}}$, and time cost $v^{i,d}_{\text{cost}}$. Among all the implementations of all the ciphers, the minimums of ROM, RAM and cost are also determined(possibly each from different implementations). Denote each minimum by m^d_{ROM}, m^d_{RAM}, and m^d_{cost}. Then, the performance parameter p_d for a cipher on a device d is defined by

$$p_d = \min_i \left(v^{i,d}_{\text{ROM}}/m^d_{\text{ROM}} + v^{i,d}_{\text{RAM}}/m^d_{\text{RAM}} + v^{i,d}_{\text{cost}}/m^d_{\text{cost}} \right).$$

Finally, the figure of merit for a cipher is defined by the average of three p_d's,

$$\text{FOM} = (p_{\text{AVR}} + p_{\text{MSP}} + p_{\text{ARM}}) \times \tfrac{1}{3}.$$

The definition indicates the smaller FOM is better.[8]

B.4 Usage Scenarios

A block cipher suite usually consists of three distinctive algorithms: the key schedule, encryption and decryption. However, in lightweight applications, decryption tends to lose its role due to well-designed modes of operations for block ciphers. The combined performance of the key schedule and encryption is somewhat sensitive to their usage scenarios. For an easy comparison of our cipher with the results in the literature, we adopt two scenarios: simple encryption with a fixed key and data communication with variable keys.

Fixed-key scenario: In this scenario, a cipher is used for authenticating devices. There are no key schedules or decryption steps. Round keys are fixed in the device, i.e., specifically placed in the code area. Hence, their size is added to the code size. This scenario is used in Table 6.

Communication scenario: In this scenario, a cipher is assumed to be used for data communication. It is defined as Scenario 1 in the FELICS [27]. Originally, the scenario contains encryption, decryption together with their key schedules, where 128 bytes of data are encrypted and decrypted in the CBC mode. Since encryption part is more important for lightweight application, we define *one-way* communication scenario by omitting the decryption part, which is used in Table 7. The scenario in its original meaning is also used in Table 8.

[8] It can be pointed out that the definition of the FOM has a drawback that whenever a new minimum is found by a better implementation of any cipher, the whole FOMs of all ciphers should be updated.

C Cryptanalysis Results

C.1 Tables of characteristics for (RK) Differential, Linear, and (RK) Boomerang Cryptanalysis

Tables 10, 11, and 12 show characteristics for (RK) Differential, Linear, and (RK) Boomerang Cryptanalysis, respectively.

Table 10. The best (RK) differential characteristics found. Only the input and output differences are shown. We assume that every operation is independent. The subscript $\mathtt{x}$ indicates a hexadecimal expression.

model	n/k	**round**/p	**characteristic**
single-key model	64/128	36 /2^{-63}	$(\mathtt{0004}_\mathtt{x}, \mathtt{0408}_\mathtt{x}, \mathtt{0A00}_\mathtt{x}, \mathtt{0000}_\mathtt{x}) \rightarrow (\mathtt{0005}_\mathtt{x}, \mathtt{8502}_\mathtt{x}, \mathtt{0004}_\mathtt{x}, \mathtt{0A00}_\mathtt{x})$
	128/*	$45/2^{-125}$	$(\mathtt{01028008}_\mathtt{x}, \mathtt{08200080}_\mathtt{x}, \mathtt{04000040}_\mathtt{x}, \mathtt{42040020}_\mathtt{x}) \rightarrow (\mathtt{00000000}_\mathtt{x}, \mathtt{00110004}_\mathtt{x}, \mathtt{04089102}_\mathtt{x}, \mathtt{00080010}_\mathtt{x})$
related-key model	64/128	key diff.	$(\mathtt{0000}_\mathtt{x}, \mathtt{0000}_\mathtt{x}, \mathtt{0000}_\mathtt{x}, \mathtt{0000}_\mathtt{x}, \mathtt{0040}_\mathtt{x}, \mathtt{0000}_\mathtt{x}, \mathtt{0000}_\mathtt{x}, \mathtt{4000}_\mathtt{x})$,
		$34/2^{-61}$	$(\mathtt{20A0}_\mathtt{x}, \mathtt{1050}_\mathtt{x}, \mathtt{4000}_\mathtt{x}, \mathtt{2020}_\mathtt{x}) \rightarrow (\mathtt{0141}_\mathtt{x}, \mathtt{8080}_\mathtt{x}, \mathtt{4841}_\mathtt{x}, \mathtt{830A}_\mathtt{x})$
	128/128	key diff.	$(\mathtt{00000000}_\mathtt{x}, \mathtt{00000000}_\mathtt{x}, \mathtt{00000000}_\mathtt{x}, \mathtt{40000000}_\mathtt{x})$,
		$33/2^{-125}$	$(\mathtt{00410500}_\mathtt{x}, \mathtt{00210080}_\mathtt{x}, \mathtt{80102000}_\mathtt{x}, \mathtt{40002041}_\mathtt{x}) \rightarrow (\mathtt{12810001}_\mathtt{x}, \mathtt{8A500940}_\mathtt{x}, \mathtt{08001503}_\mathtt{x}, \mathtt{06000220}_\mathtt{x})$
	128/256	key diff.	$(\mathtt{00000000}_\mathtt{x}, \mathtt{00000000}_\mathtt{x}, \mathtt{00000000}_\mathtt{x}, \mathtt{00000000}_\mathtt{x}, \mathtt{00000000}_\mathtt{x}, \mathtt{40000000}_\mathtt{x}, \mathtt{00000000}_\mathtt{x}, \mathtt{00000000}_\mathtt{x})$,
		$40/2^{-127}$	$(\mathtt{42800040}_\mathtt{x}, \mathtt{20010420}_\mathtt{x}, \mathtt{00800104}_\mathtt{x}, \mathtt{08408082}_\mathtt{x}) \rightarrow (\mathtt{04405080}_\mathtt{x}, \mathtt{80040892}_\mathtt{x}, \mathtt{01000010}_\mathtt{x}, \mathtt{80a00008}_\mathtt{x})$

Table 11. The best linear approximations found. We assume that every operation is independent.

n/k	**round**	$\langle \Gamma_{\text{in}}, \Gamma_{\text{out}} \rangle$	ϵ
64/128	34	$\langle (\mathtt{0000}_\mathtt{x}, \mathtt{1000}_\mathtt{x}, \mathtt{10D9}_\mathtt{x}, \mathtt{8C20}_\mathtt{x}), (\mathtt{CF06}_\mathtt{x}, \mathtt{0202}_\mathtt{x}, \mathtt{0100}_\mathtt{x}, \mathtt{0009}_\mathtt{x}) \rangle$	2^{-31}
128/*	40	$\langle (\mathtt{00000000}_\mathtt{x}, \mathtt{08001000}_\mathtt{x}, \mathtt{40891038}_\mathtt{x}, \mathtt{3000C800}_\mathtt{x}), (\mathtt{06082000}_\mathtt{x}, \mathtt{00001001}_\mathtt{x}, \mathtt{42040030}_\mathtt{x}, \mathtt{20020000}_\mathtt{x}) \rangle$	2^{-63}

C.2 Impossible Differential Cryptanalysis and Zero-Correlation Linear Cryptanalysis

Impossible differential cryptanalysis [12] uses a differential characteristic that can never occur. A zero-correlation linear approximation [7] is the counter-part of the impossible differential characteristic in the linear cryptanalysis field. Examples of the best impossible differential characteristics and zero-correlation linear approximations as found here are given in Table 13.

Table 12. Examples of the best (RK) boomerang characteristics that we found.

model	n/k	round/p or q	characteristic
single-key model	64/128	$17/2^{-15}$	$(8200_x, 0100_x, 0001_x, 8000_x) \rightarrow (0400_x, 0004_x, 0502_x, 0088_x)$
		$18/2^{-16}$	$(8200_x, 0100_x, 0001_x, 8000_x) \rightarrow (0004_x, 0502_x, 0088_x, 0000_x)$
	128/128	$\mathbf{23}/2^{-30}$	$(08104000_x, 04002000_x, 00000020_x, 00000010_x)$ $\rightarrow (40010000_x, 20408100_x, 00000001_x, 02000080_x)$
		$\mathbf{24}/2^{-33}$	$(02000000_x, 41000000_x, 20410000_x, 10008000_x)$ $\rightarrow (00000000_x, 00040001_x, 81020400_x, 00000004_x)$
related-key model	64/128	key diff. $\mathbf{20}/2^{-15}$	$(0000_x, 0000_x, 0080_x, 0000_x, 0000_x, 8000_x, 0000_x, 0000_x)$ $(0084_x, 0000_x, 0000_x, 0000_x) \rightarrow (0400_x, 0105_x, 8080_x, 0100_x)$
		key diff. $\mathbf{21}/2^{-16}$	$(0000_x, 0000_x, 0000_x, 0000_x, 0080_x, 0000_x, 0000_x, 8000_x)$ $(0801_x, 8400_x, 0084_x, 0000_x) \rightarrow (0000_x, 0400_x, 0105_x, 8080_x)$
	128/128	key diff. $18/2^{-31}$	$(00000000_x, 00000000_x, 00000000_x, 00800000_x)$ $(00000801_x, 80000400_x, 00800004_x, 00000000_x)$ $\rightarrow (00020484_x, 01000005_x, 08020000_x, 04010A00_x)$
		key diff. $18/2^{-31}$	$(00000000_x, 00000000_x, 00000000_x, 00800000_x)$ $(00000801_x, 80000400_x, 00800004_x, 00000000_x)$ $\rightarrow (00020484_x, 01000005_x, 08020000_x, 04010A00_x)$
	128/256	key diff. $\mathbf{23}/2^{-27}$	$(00000000_x, 00000000_x, 00000000_x, 00000000_x)$ $(00000000_x, 40000000_x, 00000000_x, 00000000_x)$ $(08410002_x, 04008000_x, 00040080_x, 00020040_x)$ $\rightarrow (02020000_x, 01010240_x, 00000202_x, 04000004_x)$
		key diff. $\mathbf{24}/2^{-32}$	$(00000000_x, 00000000_x, 00000000_x, 00000000_x)$ $(00000000_x, 40000000_x, 00000000_x, 00000000_x)$ $(08410002_x, 04008000_x, 00040080_x, 00020040_x)$ $\rightarrow (01010240_x, 00000202_x, 04000004_x, 02008002_x)$

Table 13. Examples of the best impossible differential characteristics and zero correlation linear approximations found here.

type	n/k	round	characteristic
impossible differential	64/128	10	$(1^1 0^{15}, 0^{16}, 0^{16}, 0^{16}) \leftrightarrow (x^6 1^1 0^1 x^8, x^{14} 1^1 x^1, x^{16}, x^{13} 1^1 0^1 x^1)$
		8	$\not\leftrightarrow (x^{16}, x^{16}, x^{16}, x^8 1^1 0^7) \leftrightarrow (0^8 1^1 0^7, 0^{16}, 0^{16}, 0^{15} 1^1)$
	128/*	8	$(0^{32}, 0^{32}, 0^{32}, 1^1 0^{31}) \leftrightarrow (0^{32}, x^{32}, x^{32}, x^{30} 1^1 x^1)$
		7	$\not\leftrightarrow (x^{32}, x^{32}, x^{32}, x^{24} 1^1 0^7) \leftrightarrow (0^{32}, 0^{24} 1^1 0^7, 0^{32}, 0^{32})$
zero correlation linear approximation	64/128	11	$(0^{15} 1^1, 1^1 0^{15}, 0^{16}, 0^{15} 1^1) \leftrightarrow (x^{16}, x^8 0^6 1^1 x^1, x^{16}, x^{16})$
		10	$\not\leftrightarrow (1^1 x^{15}, x^8 1^1 x^7, x^{16}, x^{16}) \leftrightarrow (0^{13} 1^1 0^2, 0^{16}, 0^{16}, 0^{16})$
	128/*	9	$(0^{32}, 0^{32}, 0^{31} 1^1, 1^1 0^{31}) \leftrightarrow (x^{32}, 0^{15} 1^1 x^8 0^8, x^{32}, x^{32})$
		9	$\not\leftrightarrow (0^{32}, 1^1 x^{31}, x^{32}, x^{32}) \leftrightarrow (0^{16} 1^1 0^{15}, 0^{32}, 0^{32}, 0^{32})$

Characteristics are denoted by 4-tuples of w-bit sequences separated by comma. 0^j, 1^j, and x^j are j-bit sequences of 0, 1, and free bit, respectively.

C.3 (RK) Differential-Linear Cryptanalysis

A (RK) differential-linear approximation [14] is constructed with a short (RK) differential characteristic and a short linear approximation. A (RK) differential-linear approximation which has a correlation of $pc^2 > 2^{-n/2}$ can be used for a (RK) differential-linear attack, where p is the probability of the differential characteristic ϕ and c is the correlation of the linear approximation ψ. Examples showing how to build these (RK) differential-linear approximations are given in Table 14.

Table 14. Examples showing how to build the best (RK) differential-linear approximations found here.

model	n/k	**(RK) diff.-linear approx.**		ϕ		ψ	
		round	pc^2	**rounds**	p	**round**	c^2
Single-key model	64/128	34	2^{-31}	20	2^{-23}	14	2^{-8}
	128/*	45	2^{-63}	24	2^{-33}	21	2^{-30}
Related-key model	64/128	36	2^{-28}	22	2^{-18}	14	2^{-10}
	128/128	39	2^{-60}	17	2^{-26}	22	2^{-34}
	128/256	45	2^{-61}	23	2^{-27}	22	2^{-34}

C.4 Integral Cryptanalysis

Integral cryptanalysis [39] uses sets of chosen plaintexts of which a part is held constant and the other part varies through all possibilities. Considering ADD-balance [33], we found the following 16-round integral characteristic for all of our ciphers.

$$(\mathcal{A},\mathcal{C},\mathcal{C},\mathcal{C}) \to (\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{A}) \to (\mathcal{C},\mathcal{C},\mathcal{A},\mathcal{C}) \to (\mathcal{C},\mathcal{A},\mathcal{C},\mathcal{C}) \to (\mathcal{A},\mathcal{C},\mathcal{C},\mathcal{A}) \to (\mathcal{C},\mathcal{C},\mathcal{A},\mathcal{A}) \to (\mathcal{C},\mathcal{A},\mathcal{A},\mathcal{C})$$
$$\to (\mathcal{A},\mathcal{A},\mathcal{C},\mathcal{A}) \to (\mathcal{A},\mathcal{C},\mathcal{A},\mathcal{B}^{+}_{\lll 1}) \to (\mathcal{C},\mathcal{A},\mathcal{B}^{+}_{\lll 1},\mathcal{A}) \to (\mathcal{A},\mathcal{B}^{+}_{\lll 1},\mathcal{A},\mathcal{A}) \to (\mathcal{B}^{+}_{\lll 1},\mathcal{A},\mathcal{A},\mathcal{U})$$
$$\to (\mathcal{A},\mathcal{A},\mathcal{U},\mathcal{U}) \to (\mathcal{A},\mathcal{U},\mathcal{U},\mathcal{B}^{+}_{\lll 8}) \to (\mathcal{U},\mathcal{U},\mathcal{B}^{+}_{\lll 8},\mathcal{U}) \to (\mathcal{U},\mathcal{B}^{+}_{\lll 8},\mathcal{U},\mathcal{U}) \to (\mathcal{B}^{+}_{\lll 8},\mathcal{U},\mathcal{U},\mathcal{U})$$

$\mathcal{C}$, $\mathcal{A}$, $\mathcal{B}^{+}_{\lll l}$, $\mathcal{U}$ represent a constant word, an active word, an ADD-balanced word when rotated to the right by l bits, and an unknown word, respectively. The above 16-round distinguisher means that if the first word of a plaintext is active, which takes all w-bit values at one time, and the other words of the plaintext are constants, then the first word of the output after 16 rounds is ADD-balanced when rotated to the right by 8 bits.

The bit-based division property [53] is an improvement of the division property [52] for non S-box-based ciphers. In [50], Sun et al. improved the integral cryptanalysis result of LEA slightly by applying the bit-based division property. Based on this result and owing to the similarity between LEA and our ciphers, we expect that the bit-based division property will not seriously improve our integral cryptanalysis.

C.5 Biclique Cryptanalysis

Yanfeng et. al. [55] showed that for variants of the Feistel structure, interleaving related-key differential trails cannot construct bicliques [5]. Hence, we consider the bicliques from independent related-key differentials, as our ciphers have a variant of the type-3 generalized Feistel structure. We calculate the total complexity C_{full} for a key recovery attack with independent bicliques using the following equation,

$$C_{full} = 2^{k-2d}(C_{biclique} + C_{precomp} + C_{recomp}),$$

where $C_{biclique}$, $C_{precomp}$, and C_{recomp} denote the complexities for building-biclique, pre-computation, and re-computation, respectively. Note that a trivial biclique for each cipher can be derived easily from related-key differentials. The specific complexities are shown in Table 15. The re-check complexity of a false positive is omitted in the above equation because it is negligible.

Table 15. Complexity of the biclique cryptanalysis for each parameter.

n/k	**biclique round**	$C_{biclique}$	$C_{precomp}$	C_{recomp}	C_{full}
64/128	15	$2^{14.6}$	$2^{16.6}$	$2^{31.5}$	$2^{127.5}$
128/128	7	$2^{29.5}$	$2^{32.7}$	$2^{63.7}$	$2^{127.7}$
128/256	15	$2^{30.3}$	$2^{32.6}$	$2^{63.6}$	$2^{255.6}$

C.6 Rotational Cryptanalysis

The initial version of a rotational cryptanalysis [37] can be easily defended by constant-XOR's. However, the recently proposed rotational-XOR cryptanalysis [2] can be well-applied to ARX ciphers with constant XOR's. So, we carefully applied the rotational-XOR cryptanalysis to our algorithm and the results are shown in the Table 16. Characteristics are initial and final δ's. Refer to [2] for attack conditions and the definition of δ.

Table 16. The best rotational-XOR differential characteristics found.

n/k	**round**/p	**characteristic**(δ)
64/128	$16/2^{-62.225}$	$(0000_x, 0000_x, 0000_x, 8002_x) \rightarrow (0004_x, 020e_x, 0805_x, 0810_x)$
128/*	$23/2^{-125.545}$	$(10000004_x, 08000000_x, 02080003_x, 40040000_x)$ $\rightarrow (00100004_x, 08080400_x, 00220008_x, a1110d00_x)$

C.7 Other Attacks

Applying round constants keeps our ciphers secure against slide attacks [19]. An algebraic attack [25] is not effective for our ciphers due to the high nonlinearity of such a case.